A FFT PROGRAM FOR MICRO-COMPUTER FOR REAL-TIME APPLICATION

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

BY

K. L. CHUGH
DSS

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

AUGUST, 1978

Submitted on 17.8.78

CERTIFICATE

Certified that the work entitled 'A FFT Program for Micro-Computer for Real-Time Application' by Mr. K.L. Chugh, has been carried out under our supervision and the work has not been submitted elsewhere for a degree.

(P.R.K. Rao)
Professor,
Department of Elec. Engg.,
I.I.T. Kanpur.

(V. Rajaraman)
Professor,
Department of Electrical Engg.
and Head
Computer Centre,
I.I.T. Kanpur.

ACKNOWLEDGEMENT

I am indebted to Prof. V. Rajaraman and Prof. P.R.K. Rao for suggesting this problem and for the encouragement and guidance throughout the course of the work.

I thank Mr. U.S. Bhakat for his co-operation while working on this project.

I thank all of my friends, especially Mr. Arjun Raman, who have directly or indirectly helped me during this work.

My thanks are also due to Mr. C.M. Abraham for the excellent typing work done.

August, 1978 K.L. CHUGH

CONTENTS

CHAPTER 1

INTRODUCTION

The Fast Fourier Transform is a method for efficiently computing the discrete fourier transform of a sequence of data samples. This technique greatly reduces the number of computations required to calculate such a transform on a digital computer. Consequently, it has made feasible the use of Fourier transforms in the analysis of many problems that were previously approached by other methods. Fourier transforms are now routinely used in such diverse areas as seismic exploration, speech analysis, echo-ranging systems, vibration analysis, image processing, and many others.

Recent years have witnessed a great increase in the availability of small, relatively inexpensive computer (mini/micro-computer), which can be employed for Fast Fourier Transform of signals in real-time applications. Such computers may be used to sample incoming signals and to perform certain calculations using these data so that the results become known as the process continues. The speed of these computers may limit the number of sample points on which the FFT can be done in a specified time. This restriction can be overcome either by interfacing the special hardware with these computers or by the use of special techniques like parallel processing, memory organisation, etc.

Thus keeping in view the capability and in expensiveness of micro-computers, it was projected to develop software and Hardware for 1024 points FFT using a microprocessor.

The FFT package was designed specially to analyse data from a digital correlator for the Rake Troposcatter System. At the output of digital correlator, 10 samples (10 Normal and 10 Quadrature) are obtained in 6.2 msecs and as such 1000 samples are accumulated per file in 620 msecs. Thus the time constraint of 620 msecs for FFT of a file was fixed to achieve real-time environment. Out of this total time, most of the time is spent in computation (very small time is spent in house keeping, I/O operations and recording of the Fourier coefficients), which consists of addition and multiplication time. Since the time required for multiplication is assumed to be much greater than that for addition, the total time for computation is approximately proportional to the multiplication time for the system.

The available microprocessor either do not have hardware multiply instruction (Intel 8080, Motorala 6800), or the one like TMS 9900 which have it, perform the multiplication slowly. Thus it is essential to have a hardware multiplier unit interfaced with the microprocessor to achieve the time constraint of real-time applications.

There are four different ways : (a) Sequential processor, (b) Cascade processor, (c) Parallel iterative processor and (d) Array analyzer, in which FFT processor can be organized [5]. The sequential processor is characterized by one arithmetic unit and a computation time proportional to $\frac{N}{2}\,\mathrm{Log}_2 N$, where N is the total number of samples. The cascade processor has $\mathrm{Log}_2 N$ arithmetic units, and the computation time is proportional to $\frac{N}{2}$ in this case. The parallel iterative processor is characterized by $\frac{N}{2}$ arithmetic units and a computation time proportional to $\mathrm{Log}_2 N$. The Array analyzer is considered in which all $\frac{N}{2}\,\mathrm{Log}_2 N$ operations are performed in parallel and the execution time is simply the time required for performing one basic operation. It has $\frac{N}{2}\,\mathrm{Log}_2 N$ arithmetic units.

Evidently out of all these different organization schemes, the sequential processor is the slowest and at the same time the cheapest one, while the Array analyzer is the fastest as well as the costliest one. The execution time has been reduced in the other types of non sequential processors by introducing parallelism in arithmetic operation.

Since the sequential processor scheme is simple in its organisation and requires only one microprocessor, we considered to start with this organization. To counteract the inherent slowness of this scheme and thus to achieve the

time constraint for real-time application, a complex hardware multiplier unit (having four multipliers in it) instead of simple multiplier unit is required to be interfaced with the microprocessor.

Out of the two microprocessors (Intel 8080 and Motorala 6800) available, Intel 8080 was selected because of its better suitability for scientific calculations and available software support like cross-assembler and simulator on 7044 computer.

In this thesis, procedures are developed for implementing Fast Fourier Transform on 8080 microprocessor. Special algorithms are devised that cut down the time required to calculate FFT, thus making them useful for real-time applications.

A companion thesis by Mr. U.S. Bhakat discusses the hardware related to the implementation of FFT on microprocessor [7].

The details of FFT algorithm and their basic properties are described in the second chapter. Third chapter contains the design of FFT program and its optimization. FFT implementation on microprocessor and microprocessor limitations for real-time application are discussed in the fourth chapter. Conclusions and suggested future work are contained in the fifth chapter.

CHAPTER 2

FFT ALGORITHMS

FFT is an algorithm (i.e. a particular method of performing a series of computations) that can compute the discrete Fourier transform much more rapidly than other available algorithms. Brute force calculations required N^2 operations since N Fourier coefficients are to be evaluated and each is the sum of N products.

An algorithm developed by J.W. Cooley and J.W. Tukey reduces the computational load to $N \operatorname{Log}_B N$ where B is the base (typically a power of 2 such as 2,4,8 or 16) to which the logarithm of N is taken and also represents the number of data from the full set of N which are processed in each substep of the procedure.

There have been some variations (like Sande-Tukey algorithm) to the original algorithm and at present there are many algorithms available for calculating the FFT. We will inspect some of the variations of the basic FFT algorithm and computational structure. It is stressed that algorithms presented is not intended to be a complete set of such algorithms. Indeed, there have been many additional modification depending on the particular requirements, the limitations of available hardware, and the ingenuity of individual designer or programmer.

Most of these algorithms may be classified as either (a) in-place or (b) natural input-output [2]. In this chapter, these algorithms will be studied by the use of signal flow graph and their basic properties will be identified.

2.1 In-place Algorithm

An in-place algorithm is one in which a given component of any intermediate vector may be stored in the same location occupied by the corresponding component of the preceding vector. This type of algorithm requires less total storage, but at the same time the computational time is higher than that required for natural input-output algorithm. Other peculiar characteristic of these algorithms is either the output spectrum appears in an unnatural order or they require that the input data be arranged before entering the computation array. Thus in-place algorithms require the reordering of input or output data. This reordering process is referred as scrambling operation and is discussed next.

2.2 Scrambling Operation

The scrambled value of a given integer m will be defined as ms [2]. Assume that m can be represented in binary form as

$$m = m_{N-1}\, m_{N-2} \cdots\cdots m_1 m_o$$

The scrambled value of m is defined as

$$m_s = m_o\ m_1\ \cdots\cdots\ m_{N-2}\ m_{N-1}$$

Thus, the scrambled value of a given integer is a new number obtained by reversing the order of all the bits in the binary representation of the given number. Note that if m is scrambled twice, the original value is obtained.

For illustration, values of m and ms for N = 8 and N = 16 are given in decimal and binary form in Table 2.1.

With some of the in-place algorithms, the data must either be scrambled before or after processing. If we assume that input is in natural order, then the output requires scrambling. Then, at a particular location m, the component appearing at the output is not X(m), but rather X(ms). In this case, it would be necessary to go to location ms to obtain the component desired for the index m.

2.3 Signal Flow Graph

FFT signal flow graph for in-place algorithm for N = 8 is shown in Fig. 2.1. It consists of data array and computational arrays. Data vector or array is represented by a vertical column of nodes on the left of the graph. The vertical columns to the right of data array correspond to computational arrays and in general there will be r

N = 8

m(decimal)	0	1	2	3	4	5	6	7
m(binary)	000	001	010	011	100	101	110	111
ms(binary)	000	100	010	110	001	101	011	111
ms(decimal)	0	4	2	6	1	5	3	7

N = 16

m(decimal)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
m(binary)	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
ms(binary)	0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
ms(decimal)	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

Table 2.1

Integers and their scrambled values for N=8 and 16.

computational arrays where $r = \mathrm{Log}_2 N$.

2.4 Basic Properties of In-place Algorithm

From the flow graph, the following basic properties [4] of in-place algorithm can be identified :

i) Number of computational arrays

$$r = \mathrm{Log}_2 N$$

ii) In each computational array every node has two incoming paths and two outgoing paths

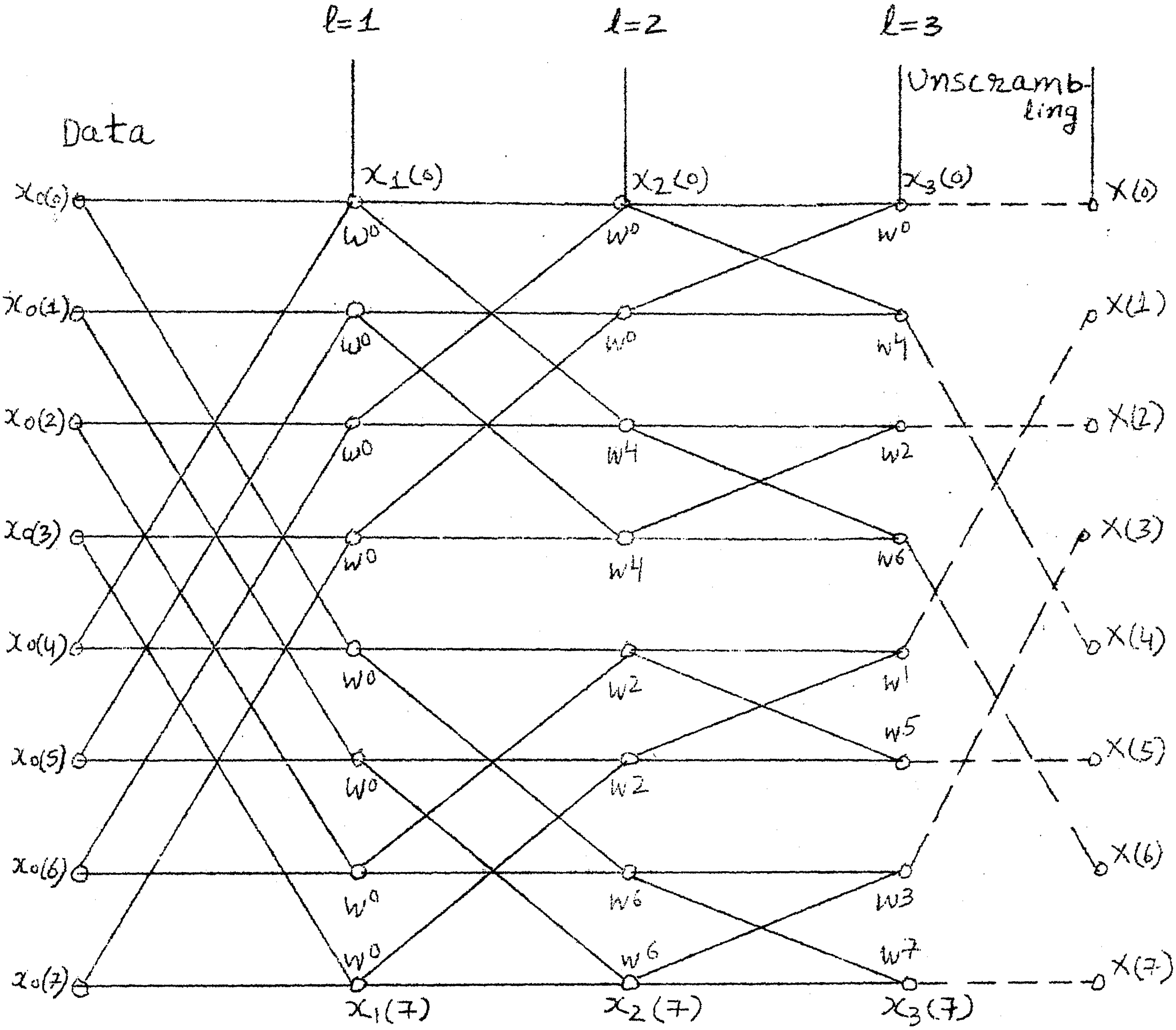
iii) In each array, there are two nodes whose input paths originate from the same pair of nodes in the previous array. Two such nodes are grouped as 'a dual node pair'. In any array there are N/2 pairs of dual nodes.

iv) Each array requires N/2 complex multiplications and N complex additions. Hence the total number of multiplications and additions required are $(N/2)\ \mathrm{Log}_2 N$ and $N\ \mathrm{Log}_2 N$ respectively. The ratio of direct to FFT computation time is

$$\frac{N^2}{N/2\ \mathrm{Log}_2 N} = \frac{2N}{\mathrm{Log}_2 N}$$

v) The computation of a dual node pair requires only one multiplication and two additions. If the weighting factor at one of the nodes in a dual node pair is W^p, then the weighting factor at the other node of the pair is $W^{p + N/2}$

Computational Arrays



Dual nodes: $x_\ell(K)$ and $x_\ell(K+N/2^\ell)$

Eight Point FFT — In-Place Algorithm

Figure 2.1

$$W^{p + N/2} = -W^{p}$$

Then

$$x_l(k) = x_{l-1}(k) + W^p x_{l-1}\ (k + N/2^l)$$

$$x_l(k + N/2^l) = x_{l-1}(k) - W^p\ x_{l-1}\ (k + N/2^l)$$

here $x_l(k)$ indicates k component in the l^{th} array.

vi) The spacing between dual node pairs differ from array to array. In the ℓ^{th} array ($\ell = 1, 2, \ldots\ldots r$) the spacing is $N/2^{\ell}$, i.e. $x\ (k)$ and $x\ (k + N/2^{\ell})$. constitute a dual node pair.

vii) To evaluate the value of P, the exponent of W, for any index in a given array, the following procedure is followed. Represent k, the node index in the ℓth array, in binary form with r bits, retain the most significant ℓ bits and add $(r - \ell)$ leading zeros to form a r bit binary number. Reverse the bit order of the resulting number. The decimal equivalent of the final binary number gives the index P. The weighting factor for the kth node of the ℓth array is W^p.

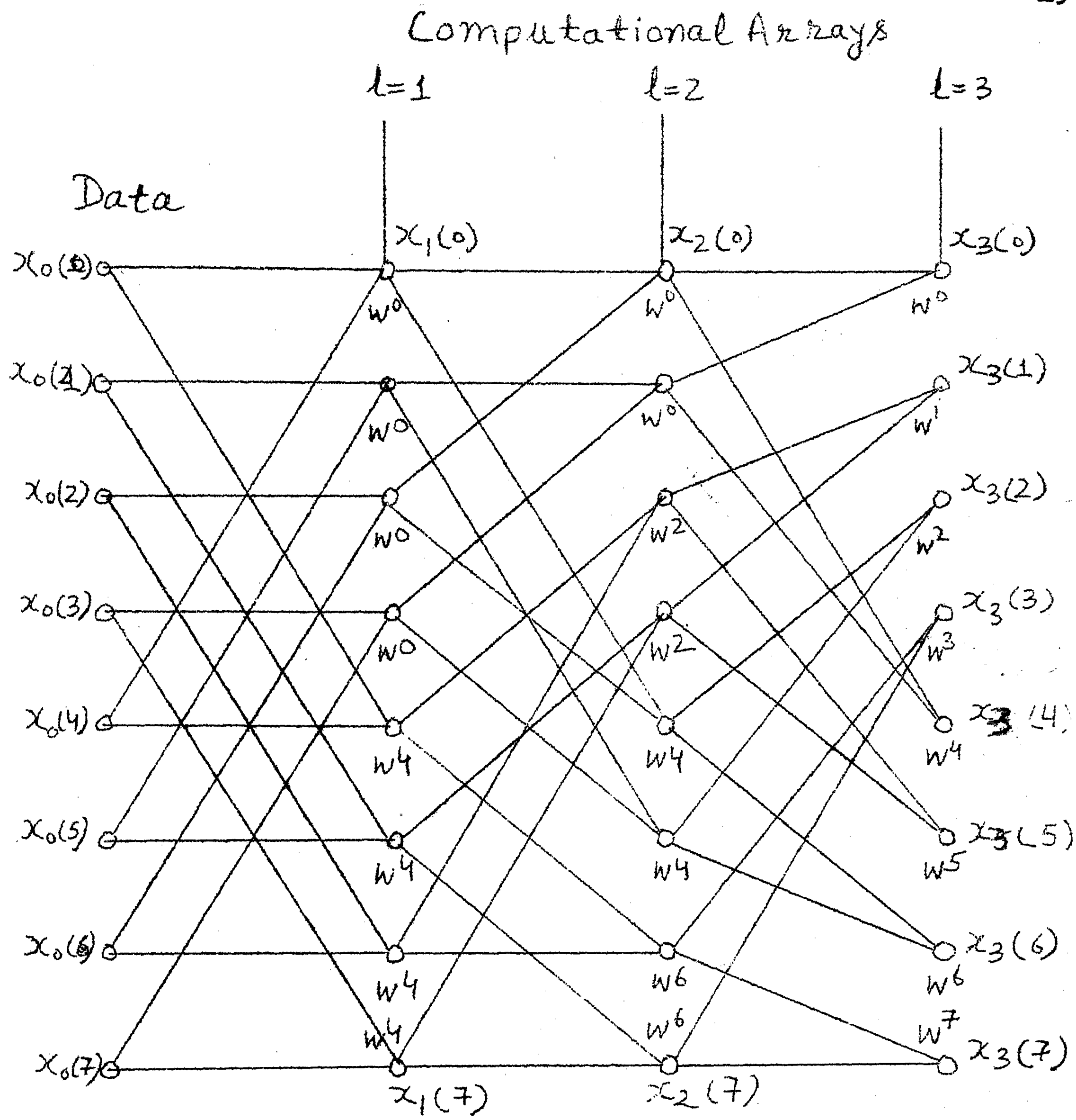
viii) The output after r arrays is in scrambled form. To unscramble the output $x(k)$, write the index k in binary form with r bits and reverse the bit order. The resulting decimal number is the index n of $x(n)$.

2.5 Natural Input-Output Algorithm

Natural input-output algorithm is one in which a given component of any intermediate vector may not be stored in the same location occupied by the corresponding component of the preceeding vector, thus requiring the extra memory for storing the intermediate result. An N point FFT (N real and N imaginary) will require 4N words of memory as compared to 2N words required for in-place algorithm. These algorithms, of course, maintain the natural input-output order and thus do not require scrambling/unscrambling at the input/output levels, and as such are faster as compared to the in-place algorithm.

2.6 Signal Flow Graph

A signal flow graph for $N = 8$ for natural input-output algorithm is given in Fig. 2.2. There are four columns and each column contain eight entries (Number of sample points). The variable $x_{\ell}(k)$ is used to denote the value of a given node in the array, where ℓ is the number of column and k is the number of the component within the column. In general ℓ varies over the range $0 \leq \ell \leq r$ with $\ell = 0$ at the left and $r = \mathrm{Log}_2 N$, and k varies over range $0 \leq k \leq N-1$ with $k = 0$ at the top.



Eight Point FFT — Unscrambling not required

(Natural Input-output Algorithm)

Figure 4.2

2.7 Basic Properties of Natural Input-Output Algorithm

Properties (i) through (iv) are same as for in-place algorithms [4].

v) The computation of a dual node pair requires only one multiplication and two additions. If the weighting factor at one of the nodes in a dual node pair is W^p, then the weighting factor at the other node of the pair is $W^{p + N/2}$

$$W^{p + N/2} = -W^p$$

Then

$$x_\ell(k) = x_{\ell-1}(i) + W^p\, x_{\ell-1}\,(i + N/2^\ell)$$

$$x_\ell(k + N/2) = x_{\ell-1}(i) - W^p\, x_{\ell-1}\,(i + N/2^\ell)$$

i depends upon location of k in the array and the number of array.

vi) The spacing between dual node pairs is same for all arrays and is N/2.

vii) To evaluate the value of P, the exponent of W, for any index in a given array, the following procedure is followed. Represent K, the node index in the ℓth array, in binary form with r bits; retain the most significant ℓ bits and add $(r - \ell)$ zeros in front of it to form a r bit binary number. The decimal equivalent of the binary number gives the index P. The weighting factor for the kth node of the ℓth array is W^p.

viii) The output after r arrays is in natural order and thus does not require unscrambling.

CHAPTER 3

FFT PROGRAM DESIGN

Programming requires a disciplined approach to the translation of requirements into unambiguous instructions for a suitable computer. However, programming involves much more than merely transcribing some symbols, it consists of at least five major steps :

i) Design

ii) Coding

iii) Translation

iv) Testing

v) Debugging

In general, if the design is not done carefully, testing and debugging will take an inordinate long time to complete. The design of a computer program requires not only the understanding of the problem, but also the suitable selection of algorithms. The selection of algorithm becomes quite critical in real-time environment, where not only the program should work , but should work efficiently i.e. should take minimum time for its execution. This requires the optimization of the program before it is tested and debugged. The optimization puts a great strain on the programmer, because he has to consider the efficiency and understandibility of the program at the same time.

This chapter will discuss the design of FFT program and its optimization for its application in real-time.

3.1 Analysis of Available FFT Program

Before starting the design of our own FFT program, it was decided to analyse the available FFT programs. After surveying the literature, only one program written in FORTRAN was found in the book 'The Fast Fourier Transform' by Brigham [1]. This program is given in Appendix 'A'. The program was run on 7044 compute for different number of sample points (1024, 2048, 4096) and execution times corresponding to these samples were calculated, by the use of Function Time (IDUM) available in FORTRAN. The results are given in Table 3.1.

It can be seen from the table that the program takes 24000 msecs. for 1024 points FFT. It is not worth trying to code this program into 8080 assembly language, since it will not be possible to achieve the time constraint of 620 msecs. for 1024 samples, however hard one may try to optimize this program.

The slowness of this program is attributed to the following reasons :

(a) The program uses in place algorithm which is slower than natural input-output algorithm.

(b) The weights W (where $W = e^{-j2\pi/N}$) are calculated by the use of library function SINE and COSINE, which consumes time and makes the program slow.

No. of sample N	Execution Time in msecs
1024	24000
4096	120000
8192	225000

Execution Time - FORTRAN Program (Brigham's Book)

Table 3.1

3.2 Selection of Algorithm

As stated above, the in-place algorithm is slow and is thus not fit for real-time environment. As such, the natural input-output algorithm is selected for the development of FFT program.

3.3 FORTRAN Program Design

The flow-chart for FFT program utilizing natural input-output algorithm is given in Fig. 3.1. FORTRAN program is developed based on this flow-chart and is given in Appendix 'B'.

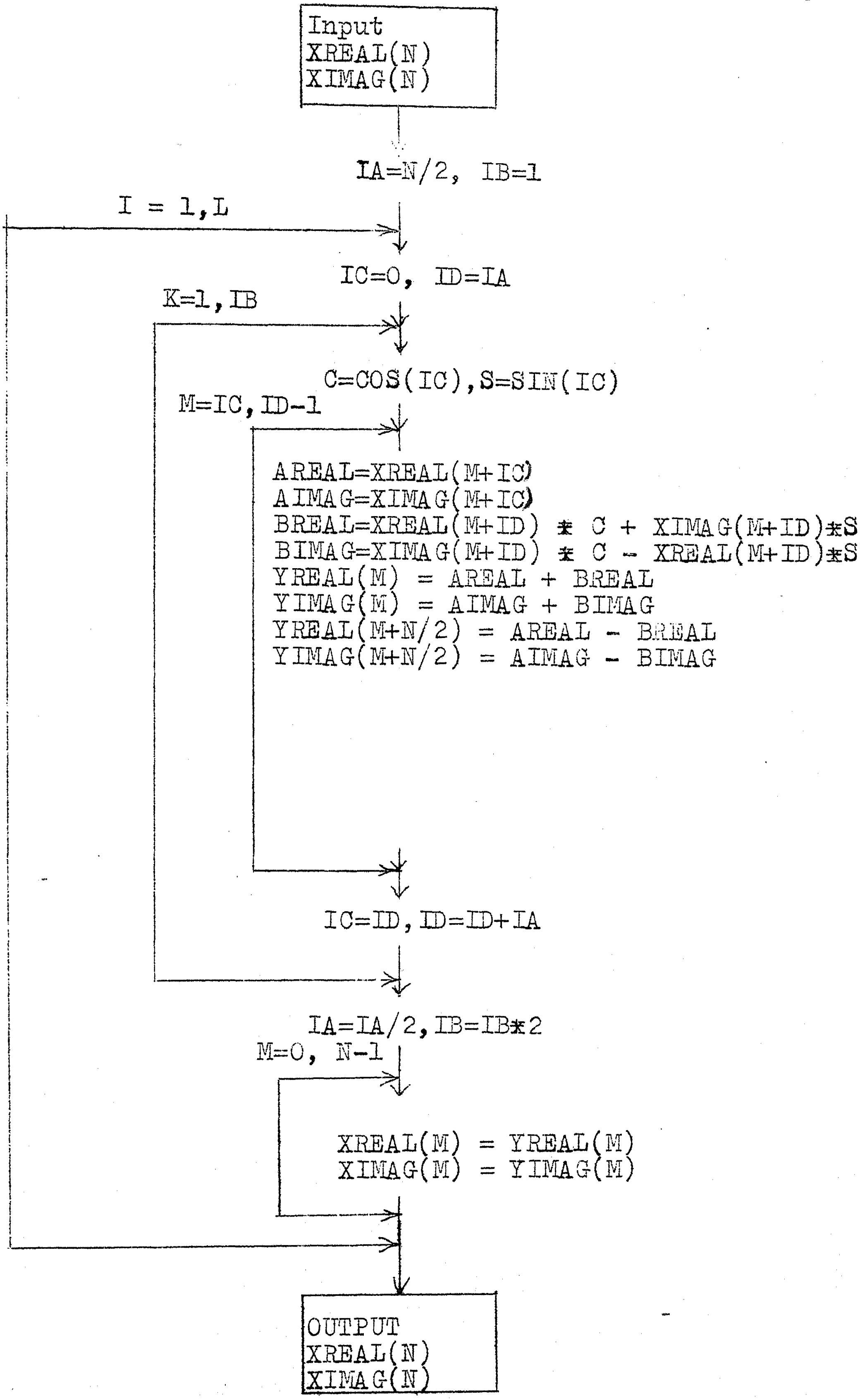


Fig. 3.1 Flow chart - FFT natural input-output

As the program is based on natural input-output algorithm, it requires more memory for storing intermediate results. For 1024 samples (1024 real and 1024 imaginary), it requires 4096 words of data storage as against 2048 words in case of in-place algorithm. YREAL and YIMAG are used for storing the intermediate results. After each iteration, YREAL and YIMAG are transferred to XREAL and XIMAG and process continues for r iterations ($r = \mathrm{Log}_2 N$). Finally, the results are stored in XREAL and XIMAG and the data which was initially in XREAL and XIMAG is lost.

The program was run on 7044 computer for 1024 sample points and it takes 3300 msec. for its execution. This time is 14 percent of the time taken by Brigham's program. The selection of natural input-output algorithm has reduced the time from 24000 msecs to 3300 msec for 1024 points FFT (86 percent reduction).

3.4 Weights of W

It is not required to call library function SINE and COSINE to calculate the weights of W (as has been done in Brigham's program), in each iteration. Values of SINE and COSINE can be generated and stored in a array before entering the FFT. N/2 values of SINE and N/2 values of COSINE are required to be stored from 0 to $\frac{2\pi}{N} * (N/2 - 1)$ in a step size of $2\pi/N$. This will correspond to N/2 weights of W, i.e.

W^o, W^1 .... $W^{N/2 - 1}$. In the program, instead of calling the library function, the index for SINE and COSINE array is calculated and appropriate values of SINE and COSINE are fetched from the array for calculation of FFT. This requires extra N words for storing the values of SINE and COSINE.

Later, in the course of development of program it was found that N/2 values of SINE and N/2 values of COSINE are not required to be stored. Only N/4 + 1 values of SINE from 0 to $\pi/2$ in a step size of $2\pi/N$ are needed and the other values of SINE and all values of COSINE can be generated from these values. These N/4 + 1 values of SINE correspond to N/4 + 1 weights of W, i.e. W^o, W^1 ..... $W^{N/4}$.

For example, for N = 1024, 257 values of SINE are stored from 0 to $\pi/2$ in step size of $2\pi/1024$ shown in Fig. 3.2. These correspond to W^o, W^1 .... W^{256}.

We will elaborate how values of SINE and COSINE are calculated from the values of SINE stored.

3.4.1 Calculation of SINE values (N = 1024)

For calculating the values of SINE required in the FFT program, following procedure is adopted :

i) The program should check whether the index is $\leq$ 256 or $>$ 256.

Location in memory	Value of sine stored
0	0
1	$2\pi/1024 * 1$
2	$2\pi/1024 * 2$
.	.
.	.
.	.
.	.
256	$2\pi/1024 * 256$ $(\pi/2)$

Values of SINE Stored in Memory

Figure 3.2

ii) If index $\leq$ 256, then use the index as it is to fetch the appropriate value of SINE from the array.

iii) If index $>$ 256, the index is subtracted from 514 (in general from 2(N/4 + 1)) and this value is used as the index for fetching the appropriate value of SINE.

3.4.2 Calculation of COSINE Values (N = 1024)

The following procedure is followed ;

i) The program checks whether the value of index $\leq$ 256 or $>$ 256 (in general, index $\leq$ N/4 or $>$ N/4).

ii) If index $\leq$ 256, subtract the index from 256 (in general, from N/4) and use this value as index to fetch the value from SINE array.

iii) If index $>$ 256, subtract 256 (in general N/4) from the index and use this value as the index to fetch the value from SINE array. This will give appropriate COSINE values.

This method of storing only (N/4 + 1) values of SINE instead of storing N/2 values of SINE and N/2 values of COSINE will cut down the memory requirement for storage of weights of W from N words to N/4 + 1 words.

The program given in Appendix 'B' was modified to incorporate the idea of weights storage instead of calling library function to calculate them. This program is given in Appendix 'C'. The program was run for 1024 samples and the

execution time was calculated. The program takes 2450 msec and this technique resulted in 26 percent reduction in execution time.

3.5 Special Loops

The program developed was further optimized by exploiting the very nature of the algorithm. There are r iterations ($r = \text{Log}_2 N$) in the FFT program. In the first iteration, the weight of W is zero, which gives value of SINE as zero and COSINE as one. This means that the components in first array can be computed by simply addition and subtraction and the need for multiplication with SINE and COSINE is eliminated. For example :

$$X_1(k)_{real} = X_o(k)_{real} + X_o(k + N/2)_{real}$$

$$X_1(k)_{imag} = X_o(k)_{imag} + X_o(k + N/2)_{imag}$$

$$X_1(k + N/2)_{real} = X_o(k)_{real} - X_o(k + N/2)_{real}$$

$$X_1(k + N/2)_{imag} = X_o(k)_{imag} - X_o(k + N/2)_{imag}$$

Similarly in second iteration, the weights of W are 0 and N/4. Zero weight of W gives values of SINE and COSINE as 0 and 1 respectively, whereas N/4 power of W will give value of SINE as 1 and COSINE as 0. As such, the components of second array can also be calculated without any multiplication.

The program was then modified such that components in array 1 (iteration 1) and array 2 (iteration 2) can be computed separately before entering the main FFT program, which will now be executed for (r - 2) number oftimes.

The modified program is given in Appendix 'D'. This technique reduced execution time for 1024 points FFT from 2450 msecs to 1900 msecs.

3.6 Memory Swapping

In natural input-output, the components calculated during the iteration are stored in the intermediate locations. These components are transferred back to their original locations before starting the next iteration. In the program YREAL and YIMAG are used as intermediate locations and XREAL and XIMAG are the original locations. As such 2N (N real and N imag) memory transfers are required after each iteration and thus a total of 2Nr ($r = \text{Log}_2 N$) memory transfers for FFT program.

These memory transfers can be avoided by the memory swapping technique. In this technique, one works on (XREAL, XIMAG) and stores the result in (YREAL, YIMAG) in the first iteration. Now instead of transferring YREAL and YIMAG to XREAL and XIMAG respectively before starting the next iteration, one has to work on (YREAL, YIMAG) and store the result in (XREAL, XIMAG) in 2nd iteration. In the next

iteration, one has to work on (XREAL, XIMAG) and store the result in (XREAL, YIMAG) and so on. Finally, the result will be in (YREAL, XIMAG) if r is even and in (YREAL, YIMAG) if r is odd.

This technique, in case of r being even, will cut down the memory transfers from 2Nr to zero and in case of r being odd, to 2N transfers only.

The technique may not save much time for large computers, in which memory transfers are quite fast, but for mini/micro computers and especially for micros, which do not have memory to memory transfer instruction, the saving in time will be appreciable.

The program was then modified to utilize this technique and resulted in execution time reduction from 1900 msecs to 1500 msecs.

The benefit gained from the use of techniques discussed 3.2 through 3.6 are shown in Table 3.2.

Thus the selection of algorithm, exploiting the very nature of algorithm and some ingenuity helped in the development of optimized program.

Technique	Execution Time for 1024 points	Time reduced to %	
Basic Brigham Program	24000 msecs	100	(reference)
Selection of algorithm (from in-place to natural)	3300 msecs	14	
Weight storage	2450 msecs	10	
Special loops	1900 msecs	8	
Memory swapping	1500 msecs	6	

Execution Time Reduction

Table 3.2

CHAPTER 4

MICROPROCESSOR IMPLEMENTATION

Ever since the development of the first microprocessor in 1971, there has been a tremendous increase in the application of microprocessor in diverse areas such as process control, instrumentation, consumer products, data acquisition and many real-time application. In most practical cases, it is realistic to consider μp for any hard wired logic employing more than 50 or 60 ICs, having more than a trivial number of steps in the flow chart and having some logical and arithmetic data processing requirement.

Eventhough microprocessors have got an advantage in their usage as compared to hard wired logic, they have their limitations in scientific and real-time application. These limitations mainly arise because of the word length and the speed of available microprocessors, which in turn dictate the available accuracy and the very utility of microprocessors in real-time application.

In this chapter, we will discuss implementation of FFT on 8080 microprocessor and microprocessor limitations for scientific real-time application, because of word length and speed.

4.1 Selection of Fixed-Point Arithmetic

To implement the FFT on 8080 microprocessor, we selected fixed-point arithmetic over floating point arithmetic, because fixed point is faster than floating point though less accurate. In the FFT program utilizing fixed point arithmetic, the input sequence (data) is scaled such that it can be represented by B bits plus sign and the binary point is assumed to lie to the left of the leftmost magnitude bit. As we move from stage to stage of the program, the magnitudes of the numbers in the sequence generally increase which means that there is possibility of incurring overflows during different stages of computations. To prevent overflows, some technique of scaling is required in fixed point arithmetic.

4.2 Scaling

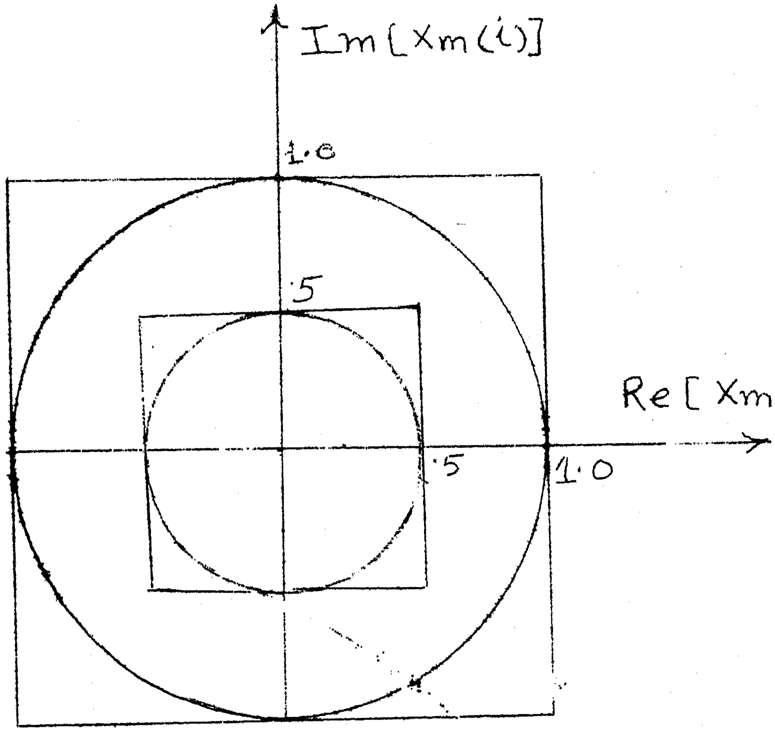
The inner loop of the power of two FFT algorithm operates on two complex numbers from the sequence [6]. It takes these two numbers and produces two new complex numbers which replace the original ones in the sequence. Let $X_m(i)$ and $X_m(j)$ be the original complex numbers. Then, the new pair $X_{m+1}(i)$, $X_{m+1}(j)$ are given by

$$X_{m+1}(i) = X_m(i) + X_m(j)\, W$$

$$X_{m+1}(j) = X_m(i) - X_m(j) W$$

At each stage, the algorithm goes through the entire sequence of N numbers in this fashion, two at a time. If $N = 2^M$, then the number of such stages in the computation is M.

With the assumption that the binary point lies at the extreme left, the relationship among the numbers in mth stage and m+1st stage is as shown in Fig. 4.1. The outside square gives the region of possible values, $Re[X_m(i)] < 1$ and $I_m[X_m(i)] < 1$. The circle inscribed in this square gives the region $|X_m(i)| < 1$. The inside square gives the region $Re[X_m(i)] < \frac{1}{2}$, $I_m[X_m(i)] < \frac{1}{2}$. Finally, the circle inscribed in this latter square gives the region $|X_m(i)| < \frac{1}{2}$. Now if $X_m(i)$ and $X_m(j)$ are inside the smaller circle, then $X_{m+1}(i)$ and $X_{m+1}(j)$ will be inside the larger circle and hence not result in an overflow. Consequently, if we control the sequence at the mth stage so that $|X_m(i)| < \frac{1}{2}$, we are certain we will have no overflow at the m+1st stage. However, if $X_m(i)$ and $X_m(j)$ are inside the smaller square, then it is possible for $X_{m+1}(i)$ or $X_{m+1}(j)$ to be outside the large square and hence result in overflow. Consequently, we can not control the sequence to prevent overflow by keeping the absolute values of the real and imaginary parts less than one-half.



Relationship Between Numbers in mth and m+1st stage

Figure 4.1

The three techniques of scaling, which when applied prevent overflow, are given below :

i) Shifting Right one Bit at Every Iteration

If the initial sequence $X_o(i)$, is scaled so that $|X_o(i)| < \frac{1}{2}$ for all i and if there is a right shift of one bit after every iteration, then there will be no overflow.

ii) Controlling the Sequence so that $|X_m(i)| < \frac{1}{2}$

Again assume the initial sequence is scaled so that $|X_o(i)| < \frac{1}{2}$ for all i. Then at each iteration we check $|X_m(i)|$ and if it is greater than one-half for any i we shift right one bit before calculation throughout the next iteration.

iii) Testing for an Overflow

In this case the initial sequence is scaled so that $Re[X_o(i)] < 1$ and $I_m[X_o(i)] < 1$. Whenever an overflow occurs in an iteration the entire sequence (part of which will be new results, part of which will be entries yet to be processed) is shifted right by one bit and the iteration is continued at the point at which the overflow occured.

The first technique is the simplest and easy to adapt for microprocessor. This method gives less accuracy than the other two techniques, since it is not generally necessary to

rescale the sequence at each iteration, there is an unnecessary loss in accuracy. The second technique requires checking $\left| X_m(i) \right|$ during each iteration and will take good amount of time if implemented on microprocessor. Since we had the time constraint, we decided not to go for this method. The third technique requires checking the overflow and as such the machine selected should have overflow as one of its status flags for its implementation. Since the 8080 microprocessor does not have an overflow flag, this technique also could not be adopted. As such, we selected the first technique of scaling to be utilized in the FFT program to be implemented on 8080 microprocessor (accuracy sacrificed for the sake of saving in execution time).

4.3 8080 Assembly Language Coding of FFT Program

The program designed in Chapter 3 was coded into assembly language of 8080 microprocessor. Since the data from digital correlator is 5 bits plus sign, we assumed that 8-bits word length of 8080 microprocessor will be enough. At this stage, we did not analyse whether 8 bits (7 bits for magnitude and one bit for sign) word length will be enough from accuracy point of view. Later simulation proved that 8-bits word length is not enough.

4.3.1 Precautions for assembly language coding

The program has got three loops a) outer loop,

b) inner loop and c) inner most loop as shown in Fig. 4.2.

The instructions in the outer loop get executed by r ($r = Log_2 N$) number of times. That is if N = 1024, the instructions in this loop will be executed by 10 times. As such this loop is not at all critical from selection of instructions to be used in this loop.

The inner loop gets executed for (N-1) times and requires some precaution in the selection of instructions which take less time for their execution.

Since there are no computation involved in outer and inner loops, the time taken for their execution will be a small proportion of the total execution time of FFT program.

The most critical of the three loops is the inner most loop. The computation of the members of the array, two at a time for power of two algorithm, is carried out in this loop. For the calculations of the entire members of the array, this loop is executed N/2 times. Since there are r arrays in the FFT program, this loop will be executed ($N/2 * r$) times. For N = 1024, this loop will be executed 5120 number of times. The execution time of this loop will be more or less equal to the execution time of the FFT program. Every care should be taken in the selection of instructions to be used in this loop.

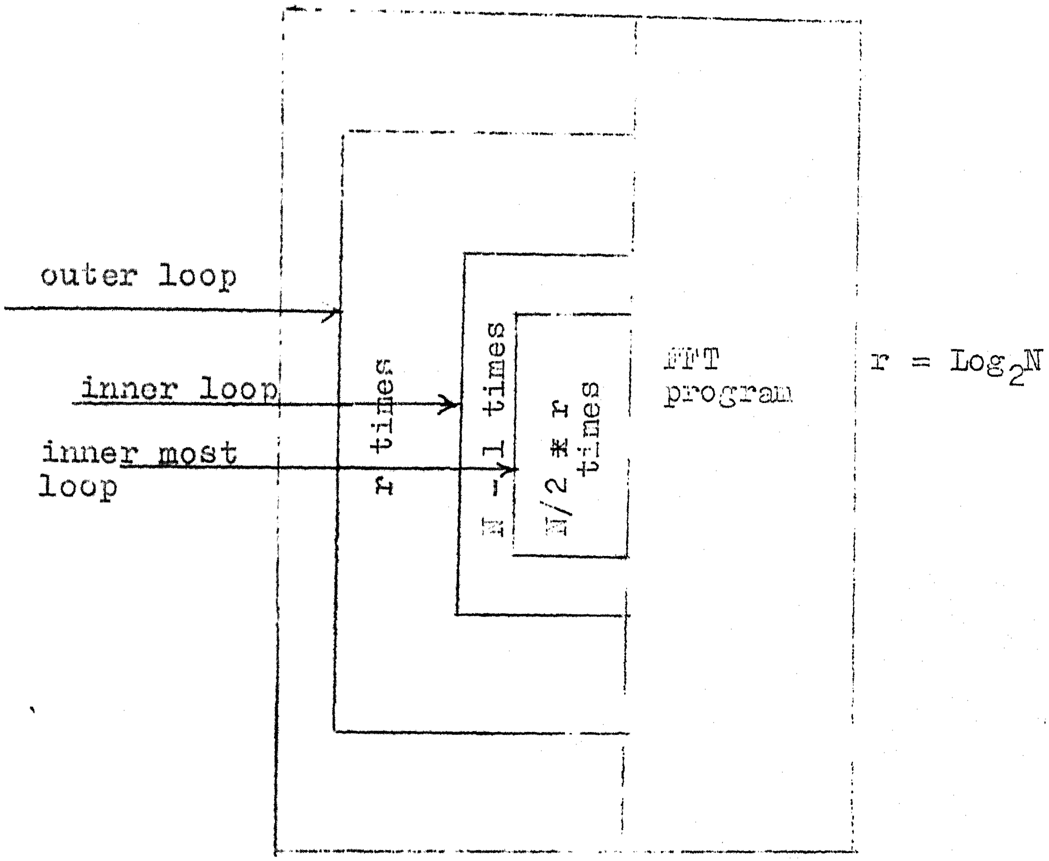


Fig. 4.2 FFT program and its loops

4.3.2 Rough Estimation of Execution Time

The execution time of each instruction (time taken from 8080 USER's MANUAL) was added in the outer, inner and inner most loops. The time calculated for each loop was multiplied by the number of times that loop gets executed and the total time, which is the sum of times taken by three loops, gave us the rough estimation of the execution time for FFT program. The time estimated was 6000 msecs. This value is a lower bound, because house keeping and input-output operations have not been included in this estimate.

4.3.3 Interfacing Hardware Multiplier

As stated above, the rough estimation of execution time gave us 6000 msecs. This time is with software multiply (8 x 8) routine, which takes 250 μsecs for 8 * 8 bits multiplication. It is clear that time target of 620 msecs for 1024 points FFT can not be achieved with software multiply routine. As such, it is required that FFT program should have hardware multiplier unit to meet the time constraint.

4.3.4 Time Estimation with Hardware Multiply

The estimation of execution time of FFT program with Hardware multiply was made. Here we assumed that a complex multiplier unit, which have 4 multiplier units, is interfaced. This multiplier can do 4 multiplications at the same

time and gives back two results after multiplication. We calculated that a minimum of 30 μsecs will be needed to perform the above operation, because 4 'OUT' instructions will be needed to send 4 operands and 2 'IN' instruction for receivers the results (each OUT/IN instruction takes 5 μsecs). The execution time with the complex hardware multiplier was calculated as 1000 msecs.

The 1000 msecs time calculation is with the available 8080 A microprocessor which has a clock frequency of 2.08 MHz. Other faster version of 8080 microprocessor, which have clock of 3.00 MHz, will give around 70 percent of the estimated time. As such we estimated that the time target of 620 msecs required for real-time application of FFT can be achieved with faster version of 8080 microprocessor interfaced with complex hardware multiplier unit and with a little optimization of the developed program.

4.4 Simulation of 8-bit Machine (microprocessor) on IBM 7044

Before going ahead with the testing of assembly language program, written using 8-bit data length, we decided to check whether 8-bits word length will be enough from accuracy point of view.

A FORTRAN program which simulates the 8-bit microprocessor on IBM 7044 computer and incorporates fixed-point arithmetic was written for Fast Fourier Transform (FFT). This was

essentially done to compare the FFT results expected with 8-bits microprocessor with fixed-point arithmetic with results obtained with 36-bits 7044 computer with floating-point arithmetic so that a decision can be taken about 8 bits word length based on the results of comparison.

4.4.1 Simulation of input data

The input data will be normalized 5 bits plus sign, i.e. it will be between 011111 and 111111, where first bit is the sign bit and other 5 bits for magnitude and the binary point is assumed to lie to the left of leftmost magnitude bit. The input sequence (Real and imaginary) was generated to lie between 0 11111 and 1 11111 to simulate the input data. This in turn means generating integers between + 124 (0 11111 00) and - 124 (1 11111 00) for 8-bit machine. For 36-bits 7044 computer, the generated input sequence will be as shown in Fig. 4.3.

4.4.2 7-Bits value of SINE

7-bits 257 values of SINE from 0 to $\pi/2$ in step size of $2\pi/1024$ were calculated. These values will vary between 0000000 and 1111111. These values of SINE were read in to the simulated program.

4.4.3 Simulation of 8-bits (7-bits for magnitude, on bit for sign) Machine

Sign bit

0 0000000000000000000000 ... 00000 1111100

28 bits 7 bits

Maximum value of data + 124

Sign bit

1 0000000000000000000000 ... 000000 1111100

28 bits 7 bits

Minimum value of data -124

Fig. 4.3 Data representation of 8-bits machine on 7044 (36-bits machine)

When two 7-bits (plus sign) numbers are multiplied, we get a result of 14 bits. Out of these 14 bits, MSB 7 bits will be used in 8-bit machine. In the simulated program, this was achieved by dividing the results of multiplication, which are 14-bits, by 2^7 i.e. equivalent of giving 7 right shifts.

4.4.4 Simulation of Fixed Point Arithmetic

As we have stated that we are going to use fixed point arithmetic in the FFT program, the magnitudes of components calculated during different stages of computations should not exceed 7-bits to prevent overflow. This can be achieved in the simulated program by ensuring that magnitude of members, calculated two at a time in the inner loop, does not exceed 127. If the absolute value of components is less than or equal to 127, the computations continue. However, when the absolute value of any component calculated exceeds 127, which is equivalent of having an overflow for 8-bits machine, the entire array is divided by two, scaling factor incremented by one and the members of that array are calculated again.

4.4.5 Conversion From Integer to Real

In the fixed-point arithmetic, the results of computations will have integer values. The integer values are converted to their real equivalent as follows :

i) Multiply the integer value by $.5/2^6$, in general by $.5/2^{B-1}$, where B is the number of bits used to represent magnitude. This aessentially means assigning weights to the integer value represented in binary. The MSB bit has a weight .5 and weight of LSB bit is $.5/2^{B-1}$.

ii) Multiply the above value by 2 to the power of scaling factor to get the final real value.

4.4.6 Results of Simulation

The simulated FFT FORTRAN program was run on IBM 7044 for 1024 sample points. The results of this program were compared with the results obtained by the FFT program using floating point arithmetic and 36-bits word length with the same input sequence data. The observations are given below :

i) Out of 1024 components (coefficients), 896 became zero i.e. 87.5 percent of the result produced by the simulated program were zero.

ii) Other 128 components, which were not zeros, were not marginally out, but differ considerably from the correct results.

The above results are not at all acceptable from accuracy point of view. Based on the results of simulation, it is concluded that 8-bits word length is not enough for FFT program employing fixed-point arithmetic.

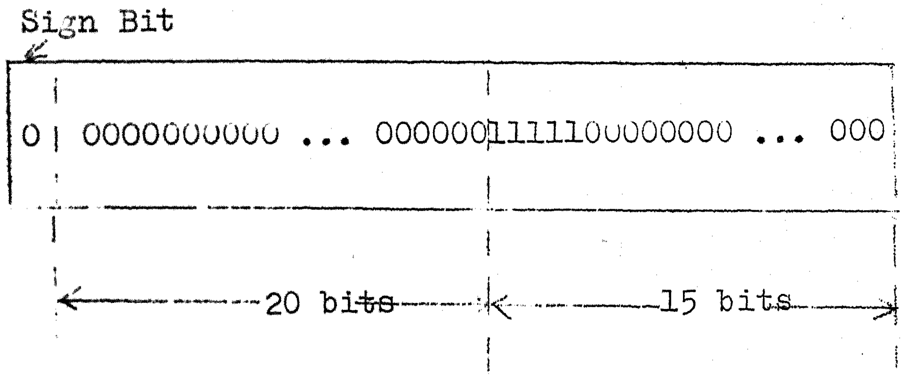
4.5 16-Bits Simulation on 7044

Since the 8-bits word length did not prove enough, the next step was to try for 16-bits word length. A FORTRAN program, which simulates 16-bits word length FFT program employing fixed point arithmetic, was written for 7044 computer. A data sequence as shown in Fig. 4.4 was generated to simulate the input data. 15-bits 257 values of SINE were calculated and read into the program. This program employs the same technique for simulation of 16-bits and fixed-point arithmetic as discussed for 8-bits simulation. The program was run on 7044 computer for 1024 sample points. The results of the program were compared with the correct results and the observations are given below :

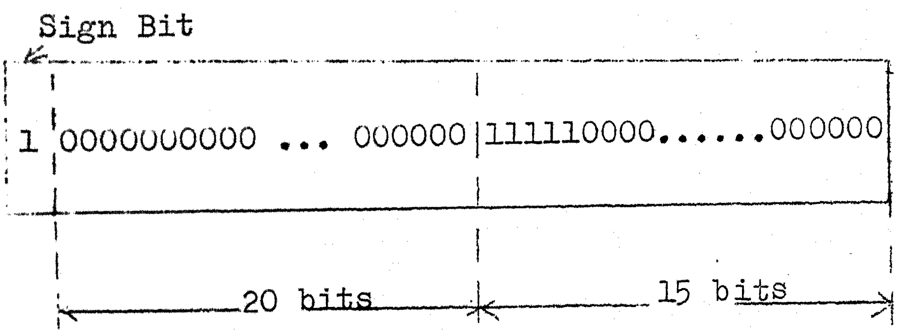
i) 99.6 percent of components were within the range of .01 to .15 from their correct values

ii) .4 percent of components differ by .16 to .3 from their correct values

Based on the above results, which are acceptable, 16-bits word length is required for FFT program employing fixed-point arithmetic. As Welch [6] has calculated for fixed-point Fast Fourier Transform, the ratio of rms error to rms result with 16-bits word length will be of the order of $2 * 10^{-3}$.



Maximum Value of Data Generated : + 31744



Minimum Value of Data Generated : -31744

Fig. 4.4 Data Representation for 16 bits machine on 7044 (36 bits machine)

4.6 Selection of 8-bits microprocessor Versus 16-bits Microprocessor

The results of 8-bits and 16-bits simulation have shown that 8-bits word length is not enough and as such 16-bits word length is required. There were two alternatives :

i) either to select 16-bits microprocessor TMS 9900 and develop FFT software for it and thus try to achieve time constraint of 620 msecs for real-time application (problem specified)

ii) or to go ahead with selected 8080 8-bits microprocessor (for which assembly language program has already been coded) and use double-precision to achieve the accuracy. With double-precision, it will not be possible to achieve the time target for real-time application.

The first alternative was not acceptable as TMS 9900 microprocessor was not available and secondly there was no software support like cross-assembler, simulator available. Without software support, it would have resulted in FFT program without being fully tested and debugged. This program could not have been used straightway in future when TMS 9900 would have been available.

The second alternative was acceptable since there is software support available for testing and debugging the assembly language program. And further, the tested program

can be run on available MDS-80 kit or Micro-78 computer, both of which use 8080 microprocessor as their processor. This will result in proven FFT package for available 8080 microprocessor based machine.

4.7 8080 Assembly Language Program with Double Precision

Assembly language program for FFT was rewritten to have double precision. In this program, the data word length is 16 bits, 15 bits for magnitude and one bit for sign. All subroutines which will be called by FFT program were redesigned to work on data word length of 16 bits. Every care was taken in their coding so that they take less amount of time for their execution.

4.7.1 Testing of Assembly Language Program

The following sub-routines which will be called by FFT program were taken first for their testing :

i) Double Precision Multiply Routine (DPMUL)

This routine multiplies two signed 16-bits numbers and produces signed 16-bits result. Negative result is represented in 2's complement.

ii) Complement Routine (COMPL)

This routine converts the signed 16-bits numbers to their 2's complement equivalent.

iii) Overflow Routine (OFL)

This routine divides the 16-bits signed number by two i.e. shifts right by one bit.

iv) Power of Two Routine (LOG2N)

It finds the $\mathrm{Log}_2 N$ of number of sample (N) assuming N is power of two.

v) Bit Reversal Routine (BITR)

It converts the positive 16-bits number to its equivalent bit reversed number. This routine is required only in case of in-place algorithm.

vi) Unscrambling Routine (UNSCM)

This routine orders the output sequence which will be in scrambled form in the case of in-place algorithm.

The above assembly language routines were tested using the cross-assembler and simulator for 8080 microprocessor available on 7044 computer with different combinations of data. Once these routines were proved correct, the different blocks of FFT program were proved for their correctness. The routines were linked with the FFT program and the complete assembly language program was run on 7044 using cross-assembler and simulator files. The program was tested for 8 sample points (N = 8) and the results obtained were compared with the actual results and were correct.

The further testing of the program was done on Micro-78 computer for different number of sample points.

4.8 Micro-78 Implementation

Micro-78 computer uses 8080 microprocessor as its processor and as such assembly language program developed can be implemented straightway on this micro computer.

For testing the assembly language program on Micro-78 for different number of sample points, it is assumed that data is available on paper tape in hexadecimal signed magnitude form. Random data (XREAL, XIMAG) were generated and punched on paper tape for N = 8, 128, 256, 512, 1024. As the actual data will be 5-bits plus sign, the data generated was between + 124 and - 124 in hexadecimal form.

4.8.1 Sequence of Operation for Testing on Micro-78

The main program calls ZEROM routine for clearing the lower bytes of XREAL and XIMAG. Lower bytes are made zeros since XREALs and XIMAGs are two bytes long and the datas read will be one byte in length. The main program then calls READ routine, which reads the data from paper tape and then stores it in the MSBs of XREALs and XIMAGs. FFT program is then called which calculates the Fourier coefficients and stores them in XREALs and XIMAGs. Finally, the main program calls INTFP routine. This routine converts the integer values of the coefficients into their reals

equivalent and prints them on to Tele-type.

The program was tested for N = 8, 128, 256, 512 and 1024 sample points and all possible bugs were removed using the ODS-78 (on-line-debugging system) of Micro-78. The results were compared with the results obtained with the simulated program run on 7044. The maximum difference between the results was of the order of .03 (for N = 1024). This error is caused due to integer to floating-point conversion, since the integers values of the coefficients obtained through Micro-78 and 7044 were some.

4.8.2 Memory Requirement for FFT Implementation on Micro-78

Memory requirement for FFT implementation consists of program area and data area as shown in Fig. 4.5.

4.8.2.1 Program Area

The program area consists of the following :

i) The FFT program takes 1620 bytes of memory and has been assembled from 6000 to 11135 (octal) locations in memory of Micro-78.

ii) The READ routine, which consists of INPUT, ZEROM and READ routines, takes 127 bytes and is assembled from 11150 to 11346 (octal) locations in memory.

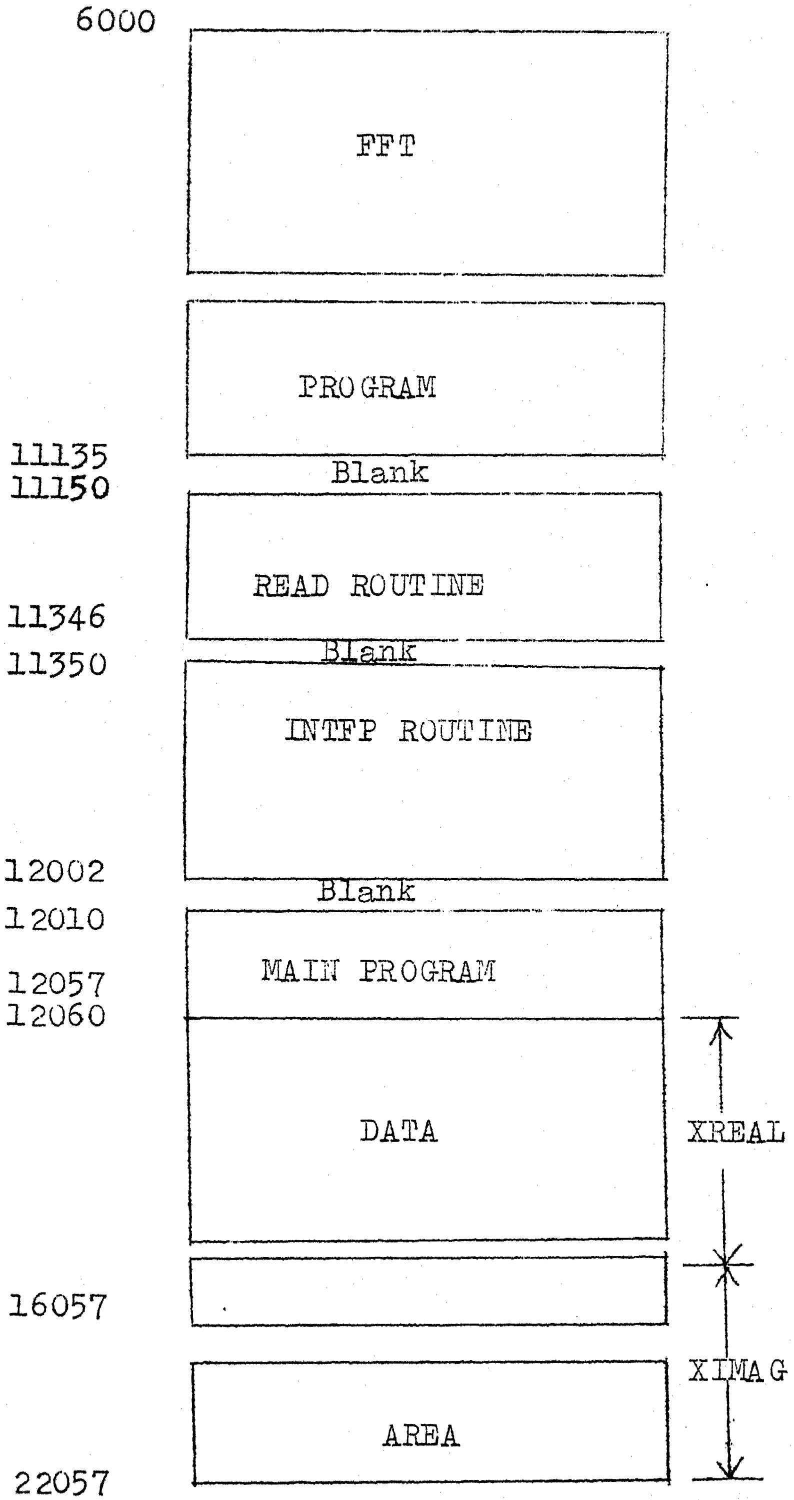


Fig. 4.5 Memory requirement for FFT implementation on Micro-78

iii) The INTEP routine, consisting of integer to floating-point conversion routine, Binary to BCD conversion routine (BNBCD) and printing routine (PRNT), takes 303 bytes and has been assembled from 11350 to 12002 (octal) locations.

iv) The MAIN program, which calls the above said routine, takes 40 bytes and is assembled from 12010 to 12057.

Thus a total of 2090 bytes of memory area has been utilized for FFT and auxiliary programs.

4.8.2.2 Data Area

As the 16-bits word length is required, 2N bytes for Real and 2N bytes for imaginary data points are required for their storage (N is number of sample points). Thus a total of 4N bytes of data area is required for FFT of N sample points.

For N = 1024, 4K bytes of data area is required and for this value of sample points, locations 12060 to 16057 have been reserved for real and locations 16060 to 22057 for imaginary data points.

4.9 Execution Time

The execution time for FFT program for different number of sample points is given in Table 4.1.

This time excludes the time taken for reading the data and the time for printing the results on the teletype.

No. of complex sample points	Time required for FFT program with software multiplication routine	Time required for FFT program with hardware (8x8 bit) multiplier
128	4 sec 500 millisec	2 sec 400 millisec
256	10 sec 500 millisec	5 sec 600 millisec
512	23 sec 200 millisec	12 sec 150 millisec
1024	51 sec 600 millisec	28 sec 300 millisec

Table 4.1

Time required for the FFT program for different complex sample points with software multiplication routine and hardware (8x8 bit) multiplier.

4.10 Limitation of Program

The complete program listing obtained through Micro-78 is given at Appendix 'F'. The program can be used for FFT of upto 1024 sample points. The limitation is because of SINE values stored in the program, which are corresponding to N = 1024 points. The weights of W (SINE and COSINE) for lower number of sample points can be calculated from these values, but for number of samples greater than 1024, this is not possible.

CHAPTER 5

CONCLUSION AND FUTURE WORK

8080 microprocessor assembly language program developed and run on micro-78 computer which uses 8080 microprocessor could not meet the real-time requirement of 1024 points FFT in 620 msecs. The time requirement could not be achieved because of 8080 being 8-bits microprocessor and the minimum requirement of 16-bits word length (proved by simulation) and also due to nonavailability of multiplier chips to build 16 * 16 bits complex multiplier unit. As it was decided to use available 8-bits 8080 microprocessor due to the nonavailability of 16 bits microprocessor and/or its related software support, double precision was to be used in the FFT program for this microprocessor. The early idea of interfacing of 8 * 8 bits complex hardware multiplier with FFT program, with which we calculated 1000 msecs for 1024 points and estimated to achieve 620 msecs with faster version of 8080, was to be abondoned. Double precision required 16 * 16 bits complex hardware multiplier unit. The 16 * 16 bits complex multiplier could not be built due to the nonavailability of multiplier chips. As such software double precision multiply routine (to multiply 16 * 16 bits number), which calls 8 * 8 bits hardware multiply unit was used in the FFT program. This resulted in 30 fold increase

in the initial calculation of 1000 msecs for 1024 points FFT and gave us 30 secs for 1024 sample points.

5.1 Conclusion

The work done has resulted in the design of optamized FFT program (Chapter 3), which can be coded in the assembly language of suitable 16-bits microprocessor like TMS 9900 to achieve the real-time environment in future. It has also proved that 8-bits word length is not enough and minimum of 16-bits word length is required for FFT program using fixed-point arithmetic (result of simulation Chapter 4). And further, the 8080 assembly language program developed will be useful in the development of FFT assembly language program for 16-bits microprocessor or faster version of 8-bits microprocessor in future. Finally, it has resulted in FFT software package for micro-78 and as such FFT upto 1024 sample points can be done on micro-computer instead on large computer.

5.2 Future Work

The following related work to FFT is suggested to improve the performance (Time wise) and flexibility of the developed FFT package :

i) In the developed FFT package, the input routine reads data from paper tape. In actual practice, this data may be required to be read from tape unit. It is suggested

that tape unit be interfaced with micro-78 computer and input routine be modified accordingly.

ii) SINE values corresponding to sample points greater than 1024 be calculated and stored in ROMs. For example, if we calculate the SINE value corresponding to 2048 sample points and store in ROM, then plugging in this ROM in memory module will make the FFT program applicable upto 2048 points. This will remove the limitation of the developed program for its use upto 1024 points, because in the developed program SINE values corresponding to 1024 points have been stored.

iii) 16 * 16 bits complex hardware multiplier be built when multiplier chips are available. This multiplier unit when interfaced with micro-78 will cut down the execution time of the FFT program considerably. With this and faster version of 8080, it may be possible to achieve real-time constraint. This will involve very little modification in the software.

iv) 16-bits microprocessor like TMS 9900 may be tried for FFT. This will eliminate the need of double precision in the program. If the software developed for this microprocessor be interfaced with hardware multiplier unit, there will be great reduction in the execution time of FFT (for real-time application).

v) Parallel processor organization instead of sequential scheme may be tried to exploit parallelism in the algorithm.

vi) Higher base algorithms (Base 4, Base 8, Base 16) may be tried instead of Base 2 algorithm to reduce computation time [1].

REFERENCES

[1] BRIGHAM, E.O., The Fast Fourier Transform, Prentice-Hall, Inc. 1974.

[2] STANLEY, W.D., Digital Signal Processing, Reston Publishing Co., Inc., Reston, 1975.

[3] HARTWELL, J.W., 'A Procedure for Implementing the Fast Fourier Transform on Small Computers', IBM Journal of Research and Development, 15, 355 (1971).

[4] RAMAKRISHNA, B.S. et al., 'Digital Signal Processing', Speech Signal Processing, Department of Electrical Engineering, IISc., Bangalore, 14 (1977).

[5] SEN, A. et al., 'A Microprogramming Approach to FFT Processing', Proceedings of the Second All India Symposium on Computer Architecture and System Design, 43 (1976).

[6] WELCH, P.D., 'A Fixed Point Fast Fourier Transform Error Analysis', IEEE Transactions on Audio and Electro-acoustics, Vol. AU-17, No.3, pp 151-157, June, 1969.

[7] BHAKAT, U.S., 'Microprocessor Based System for Computing Fast Fourier Transform', M.Tech. Thesis, 1978, E.E. Department, IIT Kanpur.

```
C       PROGRAM TAKEN FROM BRIGHAM BOOK ON FFT
C       THIS PROGRAM TAKES 24000 MSECS. FOR 1024 POINTS FFT
C       THIS PROGRAM TAKES MORE TIME BUT LESS MEMORY
        SUBROUTINE FFTINP(XREAL,XIMAG,N,NU)
        DIMENSION XREAL(N),XIMAG(N)
        N2=N/2
        NU1=NU-1
        K=0
        DO 100 L=1,NU
102     DO 101 I=1,N2
        P=IBITR(K/2**NU1,NU)
        ARG=6.283185*P/FLOAT(N)
        C=COS(ARG)
        S=SIN(ARG)
        K1=K+1
        K1N2=K1+N2
        TREAL=XREAL(K1N2)*C+XIMAG(K1N2)*S
        TIMAG=XIMAG(K1N2)*C-XREAL(K1N2)*S
        XREAL(K1N2)=XREAL(K1)-TREAL
        XIMAG(K1N2)=XIMAG(K1)-TIMAG
        XREAL(K1)=XREAL(K1)+TREAL
        XIMAG(K1)=XIMAG(K1)+TIMAG
101     K=K+1
        K=K+N2
        IF (K.LT.N) GO TO 102
        K=0
        NU1=NU1-1
100     N2=N2/2
        DO 103 K=1,N
        I=IBITR(K-1,NU)+1
        IF (I.LE.K) GO TO 103
        TREAL=XREAL(K)
        TIMAG=XIMAG(K)
        XREAL(K)=XREAL(I)
        XIMAG(K)=XIMAG(I)
        XREAL(I)=TREAL
        XIMAG(I)=TIMAG
103     CONTINUE
        RETURN
        END
        FUNCTION IBITR(J,NU)
        J1=J
        IBITR=0
        DO 200 I=1,NU
        J2=J1/2
        IBITR=IBITR*2+(J1-2*J2)
200     J1=J2
        RETURN
        END
```

```
C       NATURAL ALGORITHM WITHOUT SINE STORAGE
C       THIS PROGRAM TAKES 3300 MSEC. FOR 1024 POINTS FFT
C       THIS ALGORITHIM TAKES MORE MEMORY,BUT LESS EXECUTION TIME
C       IF TIME CONSTRAINT-USE THIS PROGRAM
C       IF MEMORY CONSTRAINT-USE IN-PLACE ALGORITHIM
C        N IS NUMBER OF SAMPLE, L=LOG2N
C       W STORES N/4+1 VALUES OF SINE ,NI=N/4+1
        SUBROUTINE FFTNAT (XREAL,XIMAG,YREAL,YIMAG,W,N,L,NI)
        DIMENSION XREAL(N),XIMAG(N),YREAL(N),YIMAG(N),W(NI)
        THETA=44.0/7.0/FLOAT(N)
        NI1=NI+1
        NI2=NI*2
        NI3=NI-1
        NBY2=N/2
        IA=N/2
        IB=1
C       OUTER LOOP STARTS
        DO 100 I=1,L
        IC=0
        ID=IA
C       INNER LOOP STARTS
        DO 50 K=1,IB
        XIC=IC
        XTHETA=THETA*XIC
        C=COS(XTHETA)
        S=SIN(XTHETA)
        IC=IC+1
C       INNER MOST LOOP STARTS
17      DO 20 M=IC,ID
        NM=M+IC-1
        MI=M+ID
        MIA=M+NBY2
        BREAL=XREAL(MI)*C+XIMAG(MI)*S
        BIMAG=XIMAG(MI)*C-XREAL(MI)*S
        AREAL=XREAL(NM)
        AIMAGE=XIMAG(NM)
        YREAL(M)=AREAL+BREAL
        YIMAG(M)=AIMAGE+BIMAG
        YREAL(MIA)=AREAL-BREAL
        YIMAG(MIA)=AIMAGE-BIMAG
20      CONTINUE
        IC=ID
        ID=ID+IA
50      CONTINUE
        IA=IA/2
        IB=IB*2
        DO 30 IJ=1,N
        XREAL(IJ)=YREAL(IJ)
        XIMAG(IJ)=YIMAG(IJ)
30      CONTINUE
100     CONTINUE
        RETURN
        END
```

```
C       NATURAL ALGORITHM WITH SINE STORAGE
C       THIS PROGRAM TAKES 2450 MSECS. FOR 1024 POINTS FFT
C       THIS ALGORITHIM TAKES MORE MEMORY BUT LESS EXECUTION TIME
C       IF TIME CONSTRAINT-USE THIS PROGRAM
C       IF MEMORY CONSTRAINT-USE IN-PLACE ALGORITHIM
C        N IS NUMBER OF SAMPLE, L=LOG2N
C       W STORES N/4+1 VALUES OF SINE ,NI=N/4+1
        SUBROUTINE FFTNAT (XREAL,XIMAG,YREAL,YIMAG,W,N,L,NI)
        DIMENSION XREAL(N),XIMAG(N),YREAL(N),YIMAG(N),W(NI)
        THETA=44.0/7.0/FLOAT(N)
C       N/4+1 VALUES OF SINE GENERATED AND STORED IN ARRAY W(J)
        W(1)=0.
        DO 10 J=2,NI
        XTHETA=THETA*FLOAT(J-1)
        W(J)=SIN(XTHETA)
10      CONTINUE
        NI1=NI+1
        NI2=NI*2
        NI3=NI-1
        NBY2=N/2
        IA=N/2
        IB=1
C       OUTER LOOP STARTS
        DO 100 I=1,L
        IC=0
        ID=IA
C       INNER LOOP STARTS
        DO 50 K=1,IB
        IC=IC+1
        IF (IC.GT.NI) GO TO 16
        S=W(IC)
        KLC=NI1-IC
        C=W(KLC)
        GO TO 17
16      JIMMY=NI2-IC
        S=W(JIMMY)
        NI4=IC-NI3
        C=-W(NI4)
C       INNER MOST LOOP STARTS
17      DO 20 M=IC,ID
        NM=M+IC-1
        MI=M+ID
        MIA=M+NBY2
        BREAL=XREAL(MI)*C+XIMAG(MI)*S
        BIMAG=XIMAG(MI)*C-XREAL(MI)*S
        AREAL=XREAL(NM)
        AIMAGE=XIMAG(NM)
        YREAL(M)=AREAL+BREAL
        YIMAG(M)=AIMAGE+BIMAG
        YREAL(MIA)=AREAL-BREAL
        YIMAG(MIA)=AIMAGE-BIMAG
20      CONTINUE
        IC=ID
        ID=ID+IA
50      CONTINUE

  2650     TOTAL TIME          95383   (TIMES ARE IN MILLISECONDS)
   104     DATA STORAGE         4404      AVAILABLE CORE       8131
 DATE  082378

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
      IA=IA/2
      IB=IB*2
      DO 30 IJ=1,N
      XREAL(IJ)=YREAL(IJ)
      XIMAG(IJ)=YIMAG(IJ)
30    CONTINUE
100   CONTINUE
      RETURN
      END
```

```
C     NATURAL ALGORITHM WITH SPECIAL LOOP
C     THIS PROGRAM TAKES 1900 MSECS. FOR 1024 POINTS FFT
C     THIS ALGORITHIM TAKES MORE MEMORY,BUT LESS EXECUTION TIME
C     IF TIME CONSTRAINT-USE THIS PROGRAM
C     IF MEMORY CONSTRAINT-USE IN-PLACE ALGORITHIM
C      N IS NUMBER OF SAMPLE, L=LOG2N
C     W STORES N/4+1 VALUES OF SINE ,NI=N/4+1
      SUBROUTINE FFTNAT (XREAL,XIMAG,YREAL,YIMAG,W,N,L,NI)
      DIMENSION XREAL(N),XIMAG(N),YREAL(N),YIMAG(N),W(NI)
      NI1=NI+1
      NI2=NI*2
      NI3=NI-1
      NBY2=N/2
      NBY4=N/4
C     N/4+1 VALUES OF SINE GENERATED AND STORED IN ARRAY W(J)
      THETA=44.0/7.0/FLOAT(N*+
      W(1)=0.
      DO 10 J=2,257
      XTHETA=THETA*FLOAT(J-1)
      W(J)=SIN(XTHETA)
   10 CONTINUE
      IA=N/2
      IB=1
C     FIRST SPECIAL LOOP
      DO 220 M=1,NBY2
      MIA=M+NBY2
      AREAL=XREAL(M)
      AIMAGE=XIMAG(M)
      XREAL(M)=AREAL+XREAL(MIA)
      XIMAG(M)=AIMAGE+XIMAG(MIA)
      XREAL(MIA)=AREAL-XREAL(MIA)
      XIMAG(MIA)=AIMAGE-XIMAG(MIA)
  220 CONTINUE
     IA=IA/2
     IB=IB*2
C    SECOND SPECIAL LOOP
     DO 23 M=1,NBY4
     MIA=M+NBY2
     MIAA=M+NBY4
     AREAL=XREAL(M)
     AIMAGE=XIMAG(M)
     XREAL(M)=AREAL+XREAL(MIAA)
     XIMAG(M)=AIMAGE+XIMAG(MIAA)
     YREAL(MIA)=AREAL-XREAL(MIAA)
     YIMAG(MIA)=AIMAGE-XIMAG(MIAA)
  23 CONTINUE
     NBY44=NBY4+1
     DO 24 M=NBY44,NBY2
     MIA=M+NBY2
     MIAA=M+NBY4
     BIMAG=XREAL(MIA)
     XREAL(M)=XREAL(MIAA)+XIMAG(MIA)
     XIMAG(M)=XIMAG(MIAA)-XREAL(MIA)
     XREAL(MIA)=XREAL(MIAA)-XIMAG(MIA)
     XIMAG(MIA)=XIMAG(MIAA)+BIMAG
```

```
--------------------------------
CS -

                  000000
                  000000
                  000055
D                 000224
D                 000000
 ERRORS           000000
--------------------------------
```

```
24    CONTINUE
      NBY22=NBY2+1
      NBY24=NBY2+NBY4
      DO 25  MJ=NBY22,NBY24
      XREAL(MJ)=YREAL(MJ)
      XIMAG(MJ)=YREAL(MJ)
25    CONTINUE
      IA=IA/2
      IB=IB*2
C      BALANCE OF L-2 LOOPS STARTS
C      OUTER LOOP STARTS HERE
       LL=L-2
       DO 100  I=1,LL
       IC=0
       ID=IA
       DO 50  K=1,IB
       IC=IC+1
C      APPROPRIATE POWER OF W TO BE USED IN INNER MOST LOOP CALC
       IF (IC.GT.NI) GO TO 16
       S=W(IC)
       KLC=NI1-IC
       C=W(KLC)
       GO TO 17
16     JIMMY=NI2-IC
       S=W(JIMMY)
       NI4=IC-NI3
       C=-W(NI4)
C      INNER MOST LOOP STARTS
17     DO 21  M=IC,ID
       NM=M+IC-1
       MI=M+ID
       MIA=M+NBY2
       BREAL=XREAL(MI)*C+XIMAG(MI)*S
       BIMAG=XIMAG(MI)*C-XREAL(MI)*S
       AREAL=XREAL(NM)
       AIMAGE=XIMAG(NM)
       YREAL(M)=AREAL+BREAL
       YIMAG(M)=AIMAGE+BIMAG
       YREAL(MIA)=AREAL-BREAL
       YIMAG(MIA)=AIMAGE-BIMAG
21     CONTINUE
       IC=ID
       ID=ID+IA
50     CONTINUE
2200   IA=IA/2
       IB=IB*2
       DO 30  IJ=1,N
       XREAL(IJ)=YREAL(IJ)
       XIMAG(IJ)=YIMAG(IJ)
30     CONTINUE
100    CONTINUE
       RETURN
       END
```

```
C       NATURAL ALGORITHM WITH SPECIAL LOOP AND MEMORY SWAPPING
C       THIS PROGRAM TAKES 1500 MSECS. FOR 1024 POINTS FFT
C       THIS ALGORITHIM TAKES MORE MEMORY,BUT LESS EXECUTION TIME
C       IF TIME CONSTRAINT-USE THIS PROGRAM
C       IF MEMORY CONSTRAINT-USE IN-PLACE ALGORITHIM
C        N IS NUMBER OF SAMPLE, L=LOG2N
C       W STORES N/4+1 VALUES OF SINE .NI=N/4+1
        SUBROUTINE FFTNAT (XREAL,XIMAG,YREAL,YIMAG,W,N,L,NI)
        DIMENSION XREAL(N),XIMAG(N),YREAL(N),YIMAG(N),W(NI)
        EQUIVALENCE (ICHECK,CHECK)
        ICON=1
        NI1=NI+1
        NI2=NI*2
        NI3=NI-1
        NBY2=N/2
        NBY4=N/4
C       N/4+1 VALUES OF SINE GENERATED AND STORED IN ARRAY W(J)
        THETA=44.0/7.0/FLOAT(N<+
        W(1)=0.
        DO 10 J=2,257
        XTHETA=THETA*FLOAT(J-1)
        W(J)=SIN(XTHETA)
10      CONTINUE
        IA=N/2
        IB=1
C       FIRST SPECIAL LOOP
        DO 220 M=1,NBY2
        MIA=M+NBY2
        AREAL=XREAL(M)
        AIMAGE=XIMAG(M)
        YREAL(M)=AREAL+XREAL(MIA)
        YIMAG(M)=AIMAGE+XIMAG(MIA)
        YREAL(MIA)=AREAL-XREAL(MIA)
        XIMAG(MIA)=AIMAGE-XIMAG(MIA)
220     CONTINUE
       IA=IA/2
       IB=IB*2
C      SECOND SPECIAL LOOP
       DO 23 M=1,NBY4
       MIA=M+NBY2
       MIAA=M+NBY4
       AREAL=XREAL(M)
       AIMAGE=XIMAG(M)
       XREAL(M)=AREAL+XREAL(MIAA)
       XIMAG(M)=AIMAGE+XIMAG(MIAA)
       YREAL(MIA)=AREAL-XREAL(MIAA)
       YIMAG(MIA)=AIMAGE-XIMAG(MIAA)
23     CONTINUE
       NBY44=NBY4+1
       DO 24 M=NBY44,NBY2
       MIA=M+NBY2
       MIAA=M+NBY4
       BIMAG=XREAL(MIA)
       XREAL(M)=XREAL(MIAA)+XIMAG(MIA)
       XIMAG(M)=XIMAG(MIAA)-XREAL(MIA)
```

```
      XREAL(MIA)=XREAL(MIAA)-XIMAG(MIA)
      XIMAG(MIA)=XIMAG(MIAA)+BIMAG
24    CONTINUE
      NBY22=NBY2+1
      NBY24=NBY2+NBY4
      DO 25  MJ=NBY22,NBY24
      XREAL(MJ)=YREAL(MJ)
      XIMAG(MJ)=YREAL(MJ)
25    CONTINUE
      IA=IA/2
      IB=IB*2
C      BALANCE OF L-2 LOOPS STARTS
C      OUTER LOOP STARTS HERE
       LL=L-2
       DO 100 I=1,LL
       IC=0
       ID=IA
C      MEMORY SWAPPING TECHNIQUE APPLIED
C      IF I ODD, WORK ON (XREAL,XIMAG) AND STORE IN (YREAL,YIMAG)
C      IF I EVEN, WORK ON (YREAL,YIMAG) AND STORE IN (XRE1L,XIMAG
       CHECK=AND(I,ICON)
       IF (ICHECK.EQ.1) GO TO 15
C      INNER LOOP WHEN I EVEN
       DO 50 K=1,IB
       IC=IC+1
C      APPROPRIATE POWER OF W TO BE USED IN INNER MOST LOOP CALCU
       IF (IC.GT.NI) GO TO 16
       S=W(IC)
       KLC=NI1-IC
       C=W(KLC)
       GO TO 17
16     JIMMY=NI2-IC
       S=W(JIMMY)
       NI4=IC-NI3
       C=-W(NI4)
C      INNER MOST LOOP STARTS
17     DO 21 M=IC,ID
       NM=M+IC-1
       MI=M+ID
       MIA=M+NBY2
       BREAL=YREAL(MI)*C+YIMAG(MI)*S
       BIMAG=YIMAG(MI)*C-YREAL(MI)*S
       AREAL=YREAL(NM)
       AIMAGE=YIMAG(NM)
       XREAL(M)=AREAL+BREAL
       XIMAG(M)=AIMAGE+BIMAG
       XREAL(MIA)=AREAL-BREAL
       XIMAG(MIA)=AIMAGE-BIMAG
21     CONTINUE
       IC=ID
       ID=ID+IA
50     CONTINUE
      GO TO 2200
C      INNER LOOP WHEN I ODD
15    DO 500 K=1,IB
```

```
      IC=IC+1
      IF (IC.GT.NI) GO TO 160
      S=W(IC)
      KIC=NI1-IC
      C=W(KIC)
      GO TO 170
160   JIMMY=NI2-IC
      S=W(JIMMY)
      NI4=IC-NI3
      C=-W(NI4)
170   DO 20 M=IC,ID
      NM=M+IC-1
      MIA=M+NBY2
      MI=M+ID
500   CONTINUE
2200    IA=IA/2
        IB=IB*2
100     CONTINUE
        RETURN
        END
```

```
           FFT IN PLACE PROGRAM FOR UP TO 1Ø24 SAMPLE
                N(SAMPLES) SHOULD BE POWER OF TWO
ØØ6ØØØ                   ORG    6ØØØB
ØØ6ØØØ ØØØ   FFTIN:      NOP
ØØ6ØØ1 3Ø3               JMP    FFT
       164
       Ø17
                257 VALUES OF SINE FROM Ø TO PIE/2
               IN STEP OF  2PIE/1Ø24 STORED
ØØ6ØØ4 ØØØ   SIN  :      DW     ØB
       ØØØ
ØØ6ØØ6 311               DW     311B
       ØØØ
ØØ6Ø1Ø ~~156~~ 222           DW     622B
       ~~ØØ2~~ 001
ØØ6Ø12 133               DW     1133B
       ØØ2
ØØ6Ø14 Ø44               DW     1444B
       ØØ3
ØØ6Ø16 355               DW     1755B
       ØØ3
ØØ6Ø2Ø 266               DW     2266B
       ØØ4
ØØ6Ø22 177               DW     2577B
       ØØ5
ØØ6Ø24 11Ø               DW     311ØB
       ØØ6
ØØ6Ø26 Ø21               DW     3421B
       ØØ7
ØØ6Ø3Ø 331               DW     3731B
       ØØ7
ØØ6Ø32 242               DW     4242B
       Ø1Ø
ØØ6Ø34 153               DW     4553B
       Ø11
ØØ6Ø36 Ø63               DW     5Ø63B
       Ø12
ØØ6Ø4Ø 373               DW     5373B
       Ø12
ØØ6Ø42 3Ø4               DW     57Ø4B
       Ø13
ØØ6Ø44 214               DW     6214B
       Ø14
ØØ6Ø46 124               DW     6524B
       Ø15
ØØ6Ø5Ø Ø34               DW     7Ø34B
       Ø16
ØØ6Ø52 344               DW     7344B
       Ø16
ØØ6Ø54 253               DW     7653B
       Ø17
ØØ6Ø56 163               DW     1Ø163B
       Ø2Ø
ØØ6Ø6Ø Ø72               DW     1Ø472B
       Ø21
ØØ6Ø62 ØØ1               DW     11ØØ1B
       Ø22
ØØ6Ø64 31Ø               DW     1131ØB
```

```
                022
006066  217             DW      11617B
        023
006070  125             DW      12125B
        024
006072  034             DW      12434B
        025
006074  342             DW      12742B
        025
006076  250             DW      13250B
        026
006100  156             DW      13556B
        027
006102  063             DW      14063B
        030
006104  371             DW      14371B
        030
006106  276             DW      14676B
        031
006110  203             DW      15203B
        032
006112  107             DW      15507B
        033
006114  014             DW      16014B
        034
006116  320             DW      16320B
        034
006120  223             DW      16623B
        035
006122  127             DW      17127B
        036
006124  032             DW      17432B
        037
006126  335             DW      17735B
        037
006130  237             DW      20237B
        040
006132  142             DW      20542B
        041
006134  044             DW      21044B
        042
006136  345             DW      21345B
        042
006140  247             DW      21647B
        043
006142  147             DW      22147B
        044
006144  050             DW      22450B
        045
006146  350             DW      22750B
        045
006150  250             DW      23250B
        046
006152  150             DW      23550B
```

```
        Ø47
ØØ6154  Ø47          DW     24Ø47B
        Ø5Ø
ØØ6156  345          DW     24345B
        Ø5Ø
ØØ616Ø  244          DW     24644B
        Ø51
ØØ6162  142          DW     25142B
        Ø52
ØØ6164  Ø37          DW     25437B
        Ø53
ØØ6166  334          DW     25734B
        Ø53
ØØ617Ø  231          DW     26231B
        Ø54
ØØ6172  125          DW     26525B
        Ø55
ØØ6174  Ø21          DW     27Ø21B
        Ø56
ØØ6176  314          DW     27314B
        Ø56
ØØ62ØØ  2Ø7          DW     276Ø7B
        Ø57
ØØ62Ø2  1Ø2          DW     3Ø1Ø2B
        Ø6Ø
ØØ62Ø4  374          DW     3Ø374B
        Ø6Ø
ØØ62Ø6  265          DW     3Ø665B
        Ø61
ØØ621Ø  156          DW     31156B
        Ø62
ØØ6212  Ø47          DW     31447B
        Ø63
ØØ6214  337          DW     31737B
        Ø63
ØØ6216  227          DW     32227B
        Ø64
ØØ622Ø  116          DW     32516B
        Ø65
ØØ6222  ØØ4          DW     33ØØ4B
        Ø66
ØØ6224  272          DW     33272B
        Ø66
ØØ6226  16Ø          DW     3356ØB
        Ø67
ØØ623Ø  Ø45          DW     34Ø45B
        Ø7Ø
ØØ6232  331          DW     34331B
        Ø7Ø
ØØ6234  215          DW     34615B
        Ø71
ØØ6236  1ØØ          DW     351ØØB

        Ø72
ØØ624Ø  363          DW     35363B
```

```
        Ø72
ØØ6242  245          DW     35645B
        Ø73
ØØ6244  127          DW     36127B
        Ø74
ØØ6246  Ø1Ø          DW     3641ØB
        Ø75
ØØ625Ø  27Ø          DW     3667ØB
        Ø75
ØØ6252  15Ø          DW     3715ØB
        Ø76
ØØ6254  Ø27          DW     37427B
        Ø77
ØØ6256  3Ø6          DW     377Ø6B
        Ø77
ØØ626Ø  164          DW     4Ø164B
        1ØØ
ØØ6262  Ø41          DW     4Ø441B
        1Ø1
ØØ6264  316          DW     4Ø716B
        1Ø1
ØØ6266  172          DW     41172B
        1Ø2
ØØ627Ø  Ø46          DW     41446B
        1Ø3
ØØ6272  321          DW     41721B
        1Ø3
ØØ6274  173          DW     42173B
        1Ø4
ØØ6276  Ø44          DW     42444B
        1Ø5
ØØ63ØØ  315          DW     42715B
        1Ø5
ØØ63Ø2  165          DW     43165B
        1Ø6
ØØ63Ø4  Ø35          DW     43435B
        1Ø7
ØØ63Ø6  3Ø4          DW     437Ø4B
        1Ø7
ØØ631Ø  152          DW     44152B
        11Ø
ØØ6312  Ø17          DW     44417B
        111
ØØ6314  264          DW     44664B
        111
ØØ6316  13Ø          DW     4513ØB
        112
ØØ632Ø  373          DW     45373B
        112
ØØ6322  236          DW     45636B
        113
ØØ6324  1ØØ          DW     461ØØB
        114
ØØ6326  341          DW     46341B
        114
ØØ633Ø  2Ø1          DW     466Ø1B
```

```
        115
006332 041             DW      47041B
        116
006334 300             DW      47300B
        116
006336 136             DW      47536B
        117
006340 373             DW      47773B
        117
006342 230             DW      50230B
        120
006344 064             DW      50464B
        121
006346 317             DW      50717B
        121
006350 151             DW      51151B
        122
006352 003             DW      51403B
        123
006354 233             DW      51633B
        123
006356 063             DW      52063B
        124
006360 312             DW      52312B
        124
006362 140             DW      52540B
        125
006364 366             DW      52766B
        125
006366 212             DW      53212B
        126
006370 036             DW      53436B
        127
006372 261             DW      53661B
        127
006374 103             DW      54103B
        130
006376 324             DW      54324B
        130
006400 144             DW      54544B
        131
006402 364             DW      54764B
        131
006404 202             DW      55202B
        132
006406 020             DW      55420B
        133
006410 235             DW      55635B
        133
006412 051             DW      56051B
        134
006414 264             DW      56264B
        134
006416 076             DW      56476B
        135
006420 310             DW      56710B
```

```
        135
006422 120          DW      57120B
        136
006424 327          DW      57327B
        136
006426 136          DW      57536B
        137
006430 344          DW      57744B
        137
006432 150          DW      60150B
        140
006434 354          DW      60354B
        140
006436 157          DW      60557B
        141
006440 361          DW      60761B
        141
006442 162          DW      61162B
        142
006444 362          DW      61362B
        142
006446 161          DW      61561B
        143
006450 357          DW      61757B
        143
006452 154          DW      62154B
        144
006454 351          DW      62351B
        144
006456 144          DW      62544B
        145
006460 336          DW      62736B
        145
006462 127          DW      63127B
        146
006464 320          DW      63320B
        146
006466 107          DW      63507B
        147
006470 275          DW      63675B
        147
006472 062          DW      64062B
        150
006474 247          DW      64247B
        150
006476 032          DW      64432B
        151
006500 214          DW      64614B
        151
006502 375          DW      64775B
        151
006504 156          DW      65156B
        152
006506 335          DW      65335B
        152
006510 113          DW      65513B
        153
```

```
006512 270          DW    65670B
       153
006514 044          DW    66044B
       154
006516 217          DW    66217B
       154
006520 371          DW    66371B
       154
006522 142          DW    66542B
       155
006524 312          DW    66712B
       155
006526 061          DW    67061B
       156
006530 227          DW    67227B
       156
006532 373          DW    67373B
       156
006534 137          DW    67537B
       157
006536 302          DW    67702B
       157
006540 043          DW    70043B
       160
006542 203          DW    70203B
       160
006544 343          DW    70343B
       160
006546 101          DW    70501B
       161
006550 236          DW    70636B
       161
006552 372          DW    70772B
       161
006554 125          DW    71125B
       162
006556 257          DW    71257B
       162
006560 010          DW    71410B
       163
006562 137          DW    71537B
       163
006564 266          DW    71666B
       163
006566 013          DW    72013B
       164
006570 140          DW    72140B
       164
006572 263          DW    72263B
       164
006574 005          DW    72405B
       165
006576 126          DW    72526B
       165
006600 246          DW    72646B
```

```
       165
006602 364          DW     72764B
       165
006604 102          DW     73102B
       166
006606 216          DW     73216B
       166
006610 331          DW     73331B
       166
006612 043          DW     73443B
       167
006614 154          DW     73554B
       167
006616 264          DW     73664B
       167
006620 373          DW     73773B
       167
006622 100          DW     74100B
       170
006624 205          DW     74205B
       170
006626 310          DW     74310B
       170
006630 012          DW     74412B
       171
006632 112          DW     74512B
       171
006634 212          DW     74612B
       171
006636 311          DW     74711B
       171
006640 006          DW     75006B
       172
006642 102          DW     75102B
       172
006644 175          DW     75175B
       172
006646 267          DW     75267B
       172
006650 357          DW     75357B
       172
006652 047          DW     75447B
       173
006654 135          DW     75535B
       173
006656 222          DW     75622B
       173
006660 306          DW     75706B
       173
006662 371          DW     75771B
       173
006664 052          DW     76052B
       174
006666 132          DW     76132B
       174
006670 211          DW     76211B
```

```
      174
006672 267              DW      76267B
       174
006674 344              DW      76344B
       174
006676 017              DW      76417B
       175
006700 072              DW      76472B
       175
006702 143              DW      76543B
       175
006704 212              DW      76612B
       175
006706 261              DW      76661B
       175
006710 326              DW      76726B
       175
006712 373              DW      76773B
       175
006714 036              DW      77036B
       176
006716 077              DW      77077B
       176
006720 140              DW      77140B
       176
006722 177              DW      77177B
       176
006724 235              DW      77235B
       176
006726 272              DW      77272B
       176
006730 326              DW      77326B
       176
006732 360              DW      77360B
       176
006734 012              DW      77412B
       177
006736 042              DW      77442B
       177
006740 070              DW      77470B
       177
006742 116              DW      77516B
       177
006744 142              DW     77542B
       177
006746 165              DW      77565B
       177
006750 205              DW      77605B
       177
006752 230              DW      77630B
       177
006754 247              DW      77647B
       177
006756 265              DW      77665B
       177
006760 302              DW      77702B
```

```
       177
006762 316             DW     77716B
       177
006764 331             DW     77731B
       177
006766 342             DW     77742B
       177
006770 352             DW     77752B
       177
006772 361             DW     77761B
       177
006774 366             DW     77766B
       177
006776 372             DW     77772B
       177
007000 376             DW     77776B
       177
007002 377             DW     77777B
       177
007004 377             DW     77777B
       177
007006 000  NI2   :    DW     0B
       000
007010 000  I     :    DW     0B          ;OUTER LOOP PARAMETER
       000
007012 000  N     :    DW     0B          ;NUMBER OF SAMPLE POINTS
       000
007014 000  IA    :    DW     0B          ;INITIALLY WILL HAVE N/2
       000
007016 000  LL    :    DB     0B          ;STORES LOG2N
007017 000  HH    :    DB     0B
007020 000  CT512:     DW     1000B
       002
007022 000  S1024:     DW     0B
       000
007024 000  C1024:     DW     2000B
       004
007026 000  SI512:     DW     0B
       000
007030 000  SN512:     DW     0B
       000
007032 000  C512  :    DW     177000B
       376
PAGE   0
007034 001  IB    :    DW     1B
       000
007036 000  IC    :    DW     0B
       000
007040 000  ID    :    DW     0B
       000
007042 000  I2D   :    DW     0B
       000
007044 000  K     :    DW     0B
       000
007046 000  SINE  :    DW     0B
       000
```

```
007050 000   COSIN:  DW     0B
       000
007052 000   COUNT:  DB     0B
007053 000   FLAG1:  DB     0B
007054 000   FLAG2:  DB     0B
007055 000   FLAG :  DB     0B
007056 000   EL   :  DW     0B
       000
007060 000   EH   :  DW     0B
       000
007062 000   DL   :  DW     0B
       000
007064 000   DH   :  DW     0B
       000
007066 000   MIR  :  DW     0B
       000
007070 000   MII  :  DW     0B
       000
007072 000   NMR  :  DW     0B
       000
007074 000   NMI  :  DW     0B
       000
007076 000   XYRMI:  DW     0B
       000
007100 000   XYIMI:  DW     0B
       000
007102 000   AREAL:  DW     0B
       000
007104 000   AIMAG:  DW     0B
       000
007106 000   REAL1:  DW     0B
       000
007110 000   BREAL:  DW     0B
       000
007112 000   IMAG1:  DW     0B
       000
007114 000   BIMAG:  DW     0B
       000
007116 000   FLAG4:  DB     0B
007117 000   BFLAG:  DB     0B
007120 000   LL8  :  DB     0B
007121 257   DPMUL:  XRA    A
              DPMUL MULTIPLIES 16*16 BITS SIGNED NUMBER
              NEGATIVE NUMBER IN TWOS COMPLEMENT
              WHEN CALLED TWO NUMBERS IN (D,E) AND (H,L) REG.
              RESULT IN (H,L) REG.
007122 062           STA    FLAG1
       053
       016
007125 062           STA    FLAG2
       054
       016
007130 174           MOV    A,H
007131 346           ANI    200B
       200
007133 312           JZ     DCHEC
       156
```

```
        016
007136  076           MVI    A,1B
        001
007140  062           STA    FLAG1
        053
        016
007143  257           XRA    A
007144  175           MOV    A,L
007145  057           CMA
007146  306           ADI    1B
        001
007150  157           MOV    L,A
007151  174           MOV    A,H
007152  057           CMA
007153  316           ACI    0B
        000
007155  147           MOV    H,A
007156  172   DCHEC:  MOV    A,D
007157  346           ANI    200B
        200
007161  312           JZ     PROCD
        204
        016
007164  076           MVI    A,1B
        001
007166  062           STA    FLAG2
        054
        016
007171  257           XRA    A
007172  173           MOV    A,E
007173  057           CMA
007174  306           ADI    1B
        001
007176  137           MOV    E,A
007177  172           MOV    A,D
007200  057           CMA
007201  316           ACI    0B
        000
007203  127           MOV    D,A
007204  072   PROCD:  LDA    FLAG1
        053
        016
007207  107           MOV    B,A
007210  072           LDA    FLAG2
        054
        016
007213  250           XRA    B
007214  062           STA    FLAG
        055
        016
007217  325           PUSH   D
007220  345           PUSH   H
007221  114           MOV    C,H
007222  175           MOV    A,L
007223  315           CALL   MUL
        345
        016
```

```
007226 042             SHLD  EL
       056
       016
007231 171             MOV   A,C
007232 315             CALL  MUL
       345
       016
007235 042             SHLD  EH
       060
       016
007240 341             POP   H
007241 321             POP   D
007242 175             MOV   A,L
007243 132             MOV   E,D
007244 315             CALL  MUL
       345
       016
007247 042             SHLD  DL
       062
       016
007252 171             MOV   A,C
007253 315             CALL  MUL
       345
       016
007256 042             SHLD  DH
       064
       016
007261 052             LHLD  EH
       060
       016
007264 353             XCHG
007265 052             LHLD  DL
       062
       016
007270 031             DAD   D
007271 353             XCHG
007272 052             LHLD  EL
       056
       016
007275 154             MOV   L,H
007276 046             MVI   H,0B
       000
007300 031             DAD   D
007301 175             MOV   A,L
007302 353             XCHG
007303 132             MOV   E,D
007304 026             MVI   D,0B
       000
007306 052             LHLD  DH

       064
       016
007311 031             DAD   D
007312 051             DAD   H
007313 027             RAL
007314 175             MOV
```

```
007315 006            MVI   B,0B
       000
007317 210            ADC   B
007320 157            MOV   L,A
007321 072            LDA   FLAG
       055
       016
007324 376            CPI   0B
       000
007326 312            JZ    FINI1
       344
       016
007331 257            XRA   A
007332 175            MOV   A,L
007333 057            CMA
007334 306            ADI   1B
       001
007336 157            MOV   L,A
007337 174            MOV   A,H
007340 057            CMA
007341 316            ACI   0B
       000
007343 147            MOV   H,A
007344 311    FINI1:  RET
007345 041    MUL   : LXI   H,0B
       000
       000
007350 006            MVI   B,10B
       010
007352 026            MVI   D,0B
       000
007354 051    MULLP:  DAD   H
007355 027            RAL
007356 322            JNC   DEC
       364
       016
007361 031            DAD   D
007362 316            ACI   0B
       000
007364 005    DEC   : DCR   B
007365 302            JNZ   MULLP
       354
       016
007370 311            RET
              COMPL FINDS TWOS COMPLEMENT OF 16 BITS NUMBER
              WHEN CALLED NUMBER IN  (H,L) REG.
              RESULT IN (H,L) REG.
007377                ORG   7377B
007377 000    COMPL:  NOP
007400 257            XRA   A
007401 175            MOV   A,L
007402 057            CMA
007403 306            ADI   01B
       001
007405 157            MOV   L,A
007406 174            MOV   A,H
007407 057            CMA
```

```
007410 316          ACI   0B
       000
007412 147          MOV   H,A
007413 311          RET
           OFL DIVIDES 16 BITS SIGNED NUMBER BY 2
           WHEN CALLED NUMBER IN (H,L) REG.
           RESULT IN (H,L) REG.
007414 000 OFL  :   NOP
007415 076          MVI   A,0B
       000
007417 062          STA   FLAG4
       116
       016
007422 174          MOV   A,H
007423 346          ANI   200B
       200
007425 312          JZ    DVIDE
       040
       017
007430 076          MVI   A,1B
       001
007432 062          STA   FLAG4
       116
       016
007435 315          CALL  COMPL
       377
       016
007440 000 DVIDE:   NOP
007441 257          XRA   A
007442 174          MOV   A,H
007443 037          RAR
007444 147          MOV   H,A
007445 175          MOV   A,L
007446 037          RAR
007447 157          MOV   L,A
007450 072          LDA   FLAG4
       116
       016
007453 376          CPI   1B
       001
007455 372          JM    OVER
       063
       017
007460 315          CALL  COMPL
       377
       016
007463 311 OVER :   RET
           BITR CALCULATES BIT REVERSED NUMBER OF 16 BITS NUMBER
           WHEN CALLED NUMBER IN(H,L) REG.
           RESULT IN (H,L) REG.
007464 000 BITR :   NOP
007465 353          XCHG
007466 072          LDA   BFLAG
       117
       016
007471 376          CPI   1B
```

```
        001
007473  372          JM     LLLE8
        137
        017
007476  257  LLGT8:  XRA    A
007477  173          MOV    A,E
007500  036          MVI    E,10B
        010
007502  041          LXI    H,00B
        000
        000
007505  051  BRLP3:  DAD    H
007506  037          RAR
007507  322          JNC    BRLP4
        113
        017
007512  043          INX    H
007513  035  BRLP4:  DCR    E
007514  302          JNZ    BRLP3
        105
        017
007517  072          LDA    LL8
        120
        016
007522  137          MOV    E,A
007523  172          MOV    A,D
007524  051  BRLP5:  DAD    H
007525  037          RAR
007526  322          JNC    BRLP6
        132
        017
007531  043          INX    H
007532  035  BRLP6:  DCR    E
007533  302          JNZ    BRLP5
        124
        017
007536  311          RET
007537  000  LLLE8:  NOP
007540  257          XRA    A
007541  041          LXI    H,LL
        016
        016
007544  173          MOV    A,E
007545  136          MOV    E,M
007546  041          LXI    H,0B
        000
        000
007551  051  BRLP1:  DAD    H
007552  037          RAR
007553  322          JNC    BRLP2
        157
        017
007556  043          INX    H
007557  035  BRLP2:  DCR    E
007560  302          JNZ    BRLP1
        151
```

```
       Ø17
007563 311           RET
         FFT PROGRAM STARTS FROM HERE
007564 Ø52   FFT  :  LHLD   N
       Ø12
       Ø16
007567 315           CALL LOG2N
       ~~Ø56~~ io3
       Ø22
007572 257           XRA    A
PA
007573 174           MOV    A,H
007574 Ø37           RAR
007575 147           MOV    H,A
007576 175           MOV    A,L
007577 Ø37           RAR
007600 157           MOV    L,A
007601 Ø42           SHLD   IA
       Ø14
       Ø16
007604 Ø42
       Ø06           SHLD   NI2
       Ø16
007607 Ø76           MVI    A,1ØB
       Ø1Ø
007611 27Ø           CMP    B
007612 372           JM     GT8     ;TO CHECK LOG2N >8OR <8
       224
       Ø17
007615 257           XRA    A
007616 Ø62           STA    BFLAG
       117
       Ø16
007621 3Ø3           JMP    LTGT8
       237
       Ø17
007624 Ø76   GT8  :  MVI    A,1B
       ØØ1
007626 Ø62           STA    BFLAG
       117
       Ø16
007631 17Ø           MOV    A,B
007632 326           SUI    1ØB
       Ø1Ø
007634 Ø62           STA    LL8
       12Ø
       Ø16
007637 17Ø   LTGT8:  MOV    A,B
007640 Ø62           STA    LL
       Ø16
       Ø16
007643 326           SUI    Ø2B
       ØØ2
007645 1Ø7           MOV    B,A
007646 257           XRA    A
007647 Ø76           MVI    A,Ø1B
```

```
        ØØ1
ØØ7651  Ø37             RAR
ØØ7652  Ø37     LOOP :  RAR
ØØ7653  ØØ5             DCR     B
ØØ7654  3Ø2             JNZ     LOOP
        252
        Ø17
ØØ7657  Ø62             STA     HH
        Ø17
        Ø16
ØØ7662  Ø52             LHLD    C1Ø24
        Ø24
        Ø16
ØØ7665  Ø21             LXI     D,SIN
        ØØ4
        Ø14
ØØ767Ø  Ø31             DAD     D
ØØ7671  Ø42             SHLD    S1Ø24
        Ø22
        Ø16
ØØ7674  Ø52             LHLD    C512
        Ø32
        Ø16
ØØ7677  353             XCHG
ØØ77ØØ  Ø41             LXI     H,SIN
        ØØ4
        Ø14
ØØ77Ø3  Ø31             DAD     D
ØØ77Ø4  Ø42             SHLD    SI512
        Ø26
        Ø16
ØØ77Ø7  Ø52             LHLD    CT512
        Ø2Ø
        Ø16
ØØ7712  Ø21             LXI     D,SIN
        ØØ4
        Ø14
ØØ7715  Ø31             DAD     D
ØØ7716  Ø42             SHLD    SN512
        Ø3Ø
        Ø16
ØØ7721  257             XRA     A
ØØ7722  Ø62             STA     I
        Ø1Ø
        Ø16
ØØ7725  Ø41             LXI     H,Ø1B
        ØØ1
        ØØØ
ØØ773Ø  Ø42             SHLD    IB
        Ø34
        Ø16
                OUTER LOOP STARTS
ØØ7733  Ø41     OUTER:  LXI     H,ØB
        ØØØ
        ØØØ
```

```
007736 042         SHLD  IC
       036
       016
007741 042         SHLD  K
       044
       016
007744 052         LHLD  IA
       014
       016
007747 042         SHLD  ID
       040
       016
007752 051         DAD   H
007753 042         SHLD  I2D
       042
007756 052 INNER:  LHLD  IA      ;INNER LOOP STARTS
       014
       016
007761 353         XCHG
007762 052         LHLD  IC
       036
       016
007765 000 ICIA :  NOP
            POWER OF W CALCULATED IN HERE
007766 257         XRA   A
007767 172         MOV   A,D
007770 037         RAR
007771 127         MOV   D,A
007772 173         MOV   A,E
007773 037         RAR
007774 137         MOV   E,A
007775 332         JC    OICIA
       011
       020
010000 174         MOV   A,H
010001 037         RAR
010002 147         MOV   H,A
010003 175         MOV   A,L
010004 037         RAR
010005 157         MOV   L,A
010006 303         JMP   ICIA
       365
       017
010011 000 OICIA:  NOP
010012 315         CALL  BITR
       064
       017
010015 257         XRA   A
010016 072         LDA   HH
       017
       016
010021 037 ICLP :  RAR
010022 376         CPI   0B
       000
010024 312         JZ    FINIS
```

```
       033
       020
010027 051          DAD   H
010030 303          JMP   ICLP
       021
       020
010033 000  FINIS:  NOP
            CHECK POWER OF W :> OR <= 256
010034 000  GL256:  NOP
010035 174          MOV   A,H
010036 376          CPI   1B
       001
010040 332          JC    LT257
       133
       020
010043 376          CPI   1B
       001
010045 302          JNZ   GT256
       056
       020
010050 175  ACHEC:  MOV   A,L
010051 376          CPI   1B
       001
010053 332          JC    LT257
       133
       020
            VALUES OF SINE AND COSINE CALCULATED FOR W>256
010056 000  GT256:  NOP
010057 051          DAD   H
010060 353          XCHG
010061 052          LHLD  S1024
       022
       016
010064 257          XRA   A
010065 175          MOV   A,L
010066 223          SUB   E
010067 157          MOV   L,A
010070 174          MOV   A,H
010071 232          SBB   D
010072 147          MOV   H,A
010073 176          MOV   A,M
010074 001          LXI   B,SINE
       046
       016
010077 002          STAX  B
010100 043          INX   H
010101 003          INX   B
010102 176          MOV   A,M
010103 002          STAX  B
010104 052          LHLD  SI512
       026
       016
010107 031          DAD   D
010110 001          LXI   B,COSIN
       050
```

```
        016
010113  257            XRA   A
010114  176            MOV   A,M
010115  057            CMA
010116  306            ADI   1B
        001
010120  002            STAX  B
010121  043            INX   H
010122  003            INX   B
010123  176            MOV   A,M
010124  057            CMA
010125  316            ACI   0B
        000
010127  002            STAX  B
010130  303            JMP   IN1
        176
        020
        VALUES OF SINE AND COSINE CALCULATED FOR POWER OF W=<256
010133  000    LT257:  NOP
010134  051            DAD   H
010135  021            LXI   D,SIN
        004
        014
010140  353            XCHG
010141  031            DAD   D
010142  001            LXI   B,SINE
        046
        016
010145  176            MOV   A,M
010146  002            STAX  B
010147  043            INX   H
010150  003            INX   B
010151  176            MOV   A,M
010152  002            STAX  B
010153  052            LHLD  SN512
        030
        016
010156  257            XRA   A
010157  175            MOV   A,L
010160  223            SUB   E
010161  157            MOV   L,A
010162  174            MOV   A,H
010163  232            SBB   D
010164  147            MOV   H,A
010165  176            MOV   A,M
010166  001            LXI   B,COSIN
        050
        016
010171  002            STAX  B
010172  043            INX   H
010173  003            INX   B
010174  176            MOV   A,M
010175  002            STAX  B
010176  000    IN1  :  NOP
010177  052            LHLD  IA
```

```
          Ø14
          Ø16
Ø1Ø2Ø2    Ø51             DAD    H
Ø1Ø2Ø3    353             XCHG
Ø1Ø2Ø4    Ø41             LXI    H,XREAL
          Ø6Ø
          Ø24
Ø1Ø2Ø7    Ø31             DAD    D
Ø1Ø21Ø    Ø42             SHLD   MIR
          Ø66
          Ø16
Ø1Ø213    Ø41             LXI    H,XIMAG
          Ø6Ø
          Ø34
Ø1Ø216    Ø31             DAD    D
Ø1Ø217    Ø42             SHLD   MII
          Ø7Ø
          Ø16
Ø1Ø222    Ø52             LHLD   IC
          Ø36
          Ø16
Ø1Ø225    Ø51             DAD    H
Ø1Ø226    115             MOV    C,L
Ø1Ø227    1Ø4             MOV    B,H
               INNER MOST LOOP STARTS
Ø1Ø23Ø    Ø52  IMOST:     LHLD   MIR      ;BREAL CALCULATED
          Ø66
          Ø16
Ø1Ø233    Ø11             DAD    B
Ø1Ø234    3Ø5             PUSH   B
Ø1Ø235    136             MOV    E,M
Ø1Ø236    Ø43             INX    H
Ø1Ø237    126             MOV    D,M
Ø1Ø24Ø    353             XCHG
Ø1Ø241    315             CALL   OFL            ;TO PREVENT OERFLOW
          Ø14
          Ø17
```

```
010244                ORG   10244B
010244 042            SHLD  XYRMI
       076
       016
010247 353            XCHG
010250 052            LHLD  COSIN
       050
       016
010253 315            CALL  DPMUL
       121
       016
010256 042            SHLD  REAL1
       106
       016
010261 301            POP   B
010262 052            LHLD  MII
       070
       016
010265 011            DAD   B
010266 136            MOV   E,M
010267 043            INX   H
010270 126            MOV   D,M
010271 353            XCHG
010272 315            CALL  OFL
       014
       017
010275 042            SHLD  XYIMI
       100
       016
010300 353            XCHG
010301 052            LHLD  SINE
       046
       016
010304 305            PUSH  B
010305 315            CALL  DPMUL
       121
       016
010310 353            XCHG
010311 052            LHLD  REAL1
       106
       016
010314 031            DAD   D
010315 042            SHLD  BREAL
       110
       016
          BIMAG CALCULATED
010320 052            LHLD  XYIMI
       100
       016
010323 353            XCHG
010324 052            LHLD  COSIN
       050
       016
010327 315            CALL  DPMUL
       121
       016
010332 042            SHLD  IMAG1
       112
       016
010335 052            LHLD  XYRMI
```

```
       076
       016
010340 353        XCHG
010341 052        LHLD  SINE
       046
       016
010344 315        CALL  DPMUL
       121
       016
010347 353        XCHG
010350 052        LHLD  IMAG1
       112
       016
010353 353        XCHG
010354 315        CALL  COMPL
       377
       016
010357 353        XCHG
010360 031        DAD   D
010361 042        SHLD  BIMAG
       114
       016
010364 301        POP   B
010365 041        LXI   H,XREAL
       060
       024
010370 011        DAD   B
010371 136        MOV   E,M
010372 043        INX   H
010373 126        MOV   D,M
010374 353        XCHG
010375 315        CALL  OFL
       014
       017
010400 042        SHLD  AREAL
       102
       016
010403 041        LXI   H,XIMAG
       060
       034
010406 011        DAD   B
010407 136        MOV   E,M
010410 043        INX   H
010411 126        MOV   D,M
010412 353        XCHG
010413 315        CALL  OFL
       014
       017
010416 042        SHLD  AIMAG
       104
       016
           SUM OF AREAL AND BREAL(REAL VALUE OF COMPONENT) CALCULATE
010421 052        LHLD  AREAL
       102
       016
010424 353        XCHG
010425 052        LHLD  BREAL
       110
       016
010430 031        DAD   D
```

```
010431 353          XCHG
010432 041          LXI   H,XREAL
       060
       024
010435 011          DAD   B
010436 163          MOV   M,E
010437 043          INX   H
010440 162          MOV   M,D
          AREAL-BREAL CALCULATED
010441 052          LHLD  AREAL
       102
       016
010444 353          XCHG
010445 052          LHLD  BREAL
       110
       016
010450 315          CALL  COMPL
       377
       016
010453 353          XCHG
010454 031          DAD   D
010455 353          XCHG
010456 052          LHLD  MIR
       066
       016
010461 011          DAD   B
010462 163          MOV   M,E
010463 043          INX   H
010464 162          MOV   M,D
010465 052          LHLD  AIMAG
       104
       016
010470 353          XCHG
010471 052          LHLD  BIMAG         ;AIMAG+BIMAG CALCULATED
       114
       016
010474 031          DAD   D
010475 353          XCHG
010476 041          LXI   H,XIMAG
       060
       034
010501 011          DAD   B
010502 163          MOV   M,E
010503 043          INX   H
010504 162          MOV   M,D
010505 052          LHLD  AIMAG         ;AIMAG-BIMAG CALCULATED
       104
       016
010510 353          XCHG
010511 052          LHLD  BIMAG
       114
       016
010514 315          CALL  COMPL
       377
       016
010517 353          XCHG
010520 031          DAD   D
010521 353          XCHG
010522 052          LHLD  MII
       070
```

```
        016
010525  011        DAD   B
010526  163        MOV   M,E
010527  043        INX   H
010530  162        MOV   M,D
010531  003        INX   B
010532  003        INX   B
010533  171        MOV   A,C
010534  041        LXI   H,I2D
        042
        016
010537  276        CMP   M
010540  302        JNZ   IMOST
        230
        020
010543  043        INX   H
010544  170        MOV   A,B
010545  276        CMP   M
010546  302        JNZ   IMOST
        230
        020
010551  052        LHLD  IA
        014
        016
010554  051        DAD   H
010555  353        XCHG
010556  052        LHLD  IC
        036
        016
010561  031        DAD   D
010562  042        SHLD  IC
        036
        016
010565  052        LHLD  ID
        040
        016
010570  031        DAD   D
010571  042        SHLD  ID
        040
        016
010574  051        DAD   H
010575  042        SHLD  I2D
        042
        016
010600  052        LHLD  K
        044
        016
010603  043        INX   H
010604  042        SHLD  K
        044
        016
010607  353        XCHG
010610  041        LXI   H,IB
        034
        016
010613  173        MOV   A,E
010614  276        CMP   M
010615  302        JNZ   INNER
        356
        017
```

```
010620 043          INX   H
010621 172          MOV   A,D
010622 276          CMP   M
010623 302          JNZ   INNER
       356
       017
010626 257          XRA   A
010627 052          LHLD  IB
       034
       016
010632 175          MOV   A,L
010633 027          RAL
010634 157          MOV   L,A
010635 174          MOV   A,H
010636 027          RAL
010637 147          MOV   H,A
010640 042          SHLD  IB
       034
       016
010643 257          XRA   A
010644 052          LHLD  IA
       014
       016
010647 174          MOV   A,H
010650 037          RAR
010651 147          MOV   H,A
010652 175          MOV   A,L
010653 037          RAR
010654 157          MOV   L,A
010655 042          SHLD  IA
       014
       016
010660 072          LDA   I
       010
       016
010663 074          INR   A
010664 062          STA   I
       010
       016
010667 041          LXI   H,LL
       016
       016
010672 276          CMP   M
010673 302          JNZ   OUTER
       333
       017
           UNSCRAMBLING OF OUTPUT
010676 001          LXI   B,0B
       000
       000
010701 151   XLP1 : MOV   L,C
010702 140          MOV   H,B
010703 305          PUSH  B
010704 315          CALL  BITR
       064
       017
010707 174          MOV   A,H
010710 270          CMP   B
010711 372          JM    LPCON
       054
```

```
       022
010714 312         JZ     CHECA
       041
       022
010717 000  EXCHG: NOP
010720 051         DAD    H
010721 042         SHLD   BITRN
       134
       022
010724 353         XCHG
010725 151         MOV    L,C
010726 140         MOV    H,B
010727 051         DAD    H
010730 115         MOV    C,L
010731 104         MOV    B,H
010732 041         LXI    H,XREAL
       060
       024
010735 031         DAD    D
010736 136         MOV    E,M
010737 043         INX    H
010740 126         MOV    D,M
010741 053         DCX    H
010742 353         XCHG
010743 042         SHLD   TREAL
       130
       022
010746 041         LXI    H,XREAL
       060
       024
010751 011         DAD    B
010752 176         MOV    A,M
010753 022         STAX   D
010754 043         INX    H
010755 023         INX    D
010756 176         MOV    A,M
010757 022         STAX   D
010760 053         DCX    H
010761 353         XCHG
010762 052         LHLD   TREAL
       130
       022
010765 175         MOV    A,L
010766 022         STAX   D
010767 023         INX    D
010770 174         MOV    A,H
010771 022         STAX   D
010772 052         LHLD   BITRN
       134
       022
010775 353         XCHG
010776 041         LXI    H,XIMAG
       060
       034
011001 031         DAD    D
011002 136         MOV    E,M
011003 043         INX    H
011004 126         MOV    D,M
011005 053         DCX    H
011006 353         XCHG
```

```
011007 042              SHLD   TIMAG
       132
       022
011012 041              LXI    H,XIMAG
       060
       034
011015 011              DAD    B
011016 176              MOV    A,M
011017 022              STAX   D
011020 043              INX    H
011021 023              INX    D
011022 176              MOV    A,M
                               011023 022              STAX   D
011024 053              DCX    H
011025 353              XCHG
011026 052              LHLD   TIMAG
       132
       022
011031 175              MOV    A,L
011032 022              STAX   D
011033 023              INX    D
011034 174              MOV    A,H
011035 022              STAX   D
011036 303              JMP    LPCON
       054
       022
011041 175     CHECA:   MOV    A,L
011042 271              CMP    C
011043 312              JZ     LPCON
       054
       022
011046 372              JM     LPCON
       054
       022
011051 303              JMP    EXCHG
       317
       021
011054 000     LPCON:   NOP
011055 301              POP    B
011056 003              INX    B
011057 052              LHLD   N
       012
       016
011062 174              MOV    A,H
011063 270              CMP    B
011064 312              JZ     SCHEC
       072
       022
011067 303              JMP    XLP1
       301
       021
011072 175     SCHEC:   MOV    A,L
011073 271              CMP    C
011074 312              JZ     XEND
       102
       022
011077 303              JMP    XLP1
       301
       021
011102 311     XEND :   RET
```

```
                LOG2N CALCULATES POWER OF 2
                NUMBER IN (H,L) REG.
                POWER OF 2 IN B REG.
011103 257      LOG2N:  XRA    A
011104 001              LXI     B,010B
       010
       000
011107 175              MOV    A,L
011110 037      POWER:  RAR
011111 332              JC     POUT
       127
       022
011114 004              INR    B
011115 015              DCR    C
011116 302              JNZ    POWER
       110
       022
011121 016              MVI    C,010B
       010
011123 174              MOV    A,H
011124 303              JMP    POWER
       110
       022
011127 311      POUT :  RET
011130 000      TREAL:  DW     0B
       000
011132 000      TIMAG:  DW     0B
       000
011134 000      BITRN:  DW     0B
       000
```

[illegible]

```
                        ORG     11150B
                INPUT ROUTINE READS A CHARACTER FROM PAPER TAPE
                IF ODD PARITY ENCOUNTERED PROGRAM STOPS
011150 076      INPUT:  MVI     A,01B
       001
011152 323              OUT     10B
       010
011154 333      INLP :  IN      10B
       010
011156 247              ANA     A
011157 362              JP      INLP
       154
       022
011162 333              IN      11B
       011
011164 376              CPI     215B     ; IF CARRIAGE RETURN CONTINUE
       215
011166 312              JZ      INPUT
       150
       022
011171 376              CPI     12B      ;LINE FEED,CONTINUE
       012
011173 312              JZ      INPUT
       150
       022
011176 376              CPI     377B     ;RUB OUT,CONTINUE
       377
011200 312              JZ      INPUT
       150
       022
011203 376              CPI     254B     ;COMA,CONTINUE
       254
011205 312              JZ      INPUT
       150
       022
011210 376              CPI     0B
       000
011212 312              JZ      INPUT    ;SPACE ,CONTINUE
       150
```

```
        022
011215  306           ADI    0B
        000
011217  342           JPO    ERROR
        022
011222  346           ANI    177B
        177
011224  376           CPI    71B
        071
011226  312           JZ     NUM
        237
        022
011231  332           JC     NUM
        237
        022
011234  326  ALPHA:   SUI    67B
        067
011236  311           RET
011237  326  NUM   :  SUI    60B
        060
011241  311           RET
011242  166  ERROR:   HLT
              INITIALAZATION OF LSB PART OF DATA TO ZERO
011243  001  ZEROM:   LXI    B,00B
        000
        000
011246  076  ZERO1:   MVI    A,00B
        000
011250  167           MOV    M,A
011251  043           INX    H
011252  043           INX    H
011253  003           INX    B
011254  021           LXI    D,N
        012
        016
011257  032           LDAX   D
011260  271           CMP    C
011261  302           JNZ    ZERO1
        246
        022
011264  023           INX    D
011265  032           LDAX   D
011266  270           CMP    B
011267  302           JNZ    ZERO1
        246
        022
011272  311           RET
              ROUTINE READS DATA FROM PAPER TAPE
011273  043  READ :   INX    H
011274  000  READ1:   NOP
011275  001           LXI    B,00B
        000
        000
011300  315  READ2:   CALL   INPUT
        150
```

```
       022
               DATA IS IN HEX
Ø11303 ØØ7                RLC
Ø11304 ØØ7                RLC
Ø11305 ØØ7                RLC
Ø11306 ØØ7                RLC
Ø11307 137                MOV   E,A
Ø11310 315                CALL  INPUT
       15Ø
       Ø22
Ø11313 263                ORA   E
Ø11314 362                JP    PLUS
       324
       Ø22
Ø11317 137     MINUS:     MOV   E,A     ;NEGATIVE NUMBERIN SIGN MAGNITUDE
Ø11320 257                XRA   A              ;CONVERTED TO TWOS COMPLIMEN
Ø11321 223                SUB   E
Ø11322 366                ORI   2ØØB
       2ØØ
Ø11324 167     PLUS :     MOV   M,A
Ø11325 Ø43                INX   H
Ø11326 Ø43                INX   H
Ø11327 ØØ3                INX   B
Ø1133Ø Ø21                LXI   D,N
       Ø12
       Ø16
Ø11333 Ø32                LDAX  D
Ø11334 271                CMP   C
Ø11335 3Ø2                JNZ   READ2
       3ØØ
       Ø22
Ø1134Ø Ø23                INX   D
Ø11341 Ø32                LDAX  D
Ø11342 27Ø                CMP   B
Ø11343 3Ø2                JNZ   READ2
       3ØØ
       Ø22
Ø11346 311                RET
Ø1135Ø                    ORG   1135ØB
               ROUTINES INTFP AND PRNT CONVERTS INTEGER
               TO ITS REAL EQUIVALENT AND PRINT ON TELETYPE
Ø1135Ø ØØØ     INT   :    DW    ØB
       ØØØ
Ø11352 ØØØ     CSTN6:     DB    ØB
Ø11353 Ø76     INTFP:     MVI   A,Ø6B   ;FORMATTING OF OUTPUT
       ØØ6
Ø11355 Ø62                STA   CSTN6
       352
       Ø22
Ø1136Ø Ø46                MVI   H,ØB
       ØØØ
Ø11362 Ø72                LDA   LL
       Ø16
       Ø16
Ø11365 157                MOV   L,A
```

```
Ø11366 Ø51              DAD    H
Ø11367 Ø21              LXI    D,FLOAT
       277
       Ø23
Ø11372 Ø31              DAD    D
Ø11373 136              MOV    E,M
Ø11374 Ø43              INX    H
Ø11375 126              MOV    D,M
Ø11376 353              XCHG
Ø11377 Ø42              SHLD   INT
       35Ø
       Ø22
Ø11402 315              CALL   CRLF      ;CARRIAGE RETURN,LINE FEED
       37Ø
       Ø23
Ø11405 Ø41              LXI    H,XREAL
       Ø6Ø
       Ø24
Ø1141Ø 315              CALL   PRNT
       Ø25
       Ø23
Ø11413 315              CALL   CRLF
       37Ø
       Ø23
Ø11416 Ø41              LXI    H,XIMAG
       Ø6Ø
       Ø34
Ø11421 315              CALL   PRNT
       Ø25
       Ø23
Ø11424 311              RET
Ø11425 ØØ1    PRNT :    LXI    B,ØB
       ØØØ
       ØØØ
Ø1143Ø 136    PLP1 :    MOV    E,M
Ø11431 Ø43              INX    H
Ø11432 126              MOV    D,M
Ø11433 345              PUSH   H
Ø11434 3Ø5              PUSH   B
Ø11435 Ø52              LHLD   INT
       35Ø
       Ø22
Ø1144Ø 315              CALL   DPMUL
       121
       Ø16
Ø11443 Ø72              LDA    FLAG
       Ø55
       Ø16
Ø11446 346              ANI    Ø1B
       ØØ1
Ø1145Ø 312              JZ     PLP2
       Ø66
       Ø23
Ø11453 315              CALL   COMPL
```

```
        377
        Ø16
Ø11456  Ø16            MVI    C,55B     ;FOR PRINTING MINUS
        Ø55
Ø1146Ø  315            CALL   CO
        355
        Ø23
Ø11463  3Ø3            JMP    PLP21
        Ø73
        Ø23
Ø11466  Ø16   PLP2 :   MVI    C,4ØB    ;PRINTING BLANK
        Ø4Ø
Ø1147Ø  315            CALL   CO
        355
        Ø23
Ø11473  257   PLP21:   XRA    A
Ø11474  Ø51            DAD    H
Ø11475  Ø51            DAD    H
Ø11476  Ø16            MVI    C,Ø21B
        Ø21
Ø115ØØ  315            CALL   BNBCD
        17Ø
        Ø23
Ø115Ø3  11Ø            MOV    C,B
Ø115Ø4  315            CALL   OUT
        325
        Ø23
Ø115Ø7  112            MOV    C,D
Ø1151Ø  315            CALL   OUT
        325
        Ø23
Ø11513  Ø16            MVI    C,56B    ;PRINT DECIMAL POINT
        Ø56
Ø11515  315            CALL   CO
        355
        Ø23
Ø1152Ø  113            MOV    C,E
Ø11521  315            CALL   OUT
        325
        Ø23
Ø11524  Ø16            MVI    C,4ØB
        Ø4Ø
Ø11526  315            CALL   CO
        355
        Ø23
Ø11531  Ø41            LXI    H,CSTN6
        352
        Ø22
Ø11534  Ø65            DCR    M
Ø11535  3Ø2            JNZ    PLP4
        145
        Ø23
Ø1154Ø  315            CALL   CRLF
        37Ø
```

```
       023
011543 066             MVI    M,06B
       006
011545 301   PLP4 :    POP    B
011546 341             POP    H
011547 043             INX    H
011550 003             INX    B
011551 021             LXI    D,N
       012
       016
011554 032             LDAX   D
011555 271             CMP    C
011556 302             JNZ    PLP1
       030
       023
011561 023             INX    D
011562 032             LDAX   D
011563 270             CMP    B
011564 302             JNZ    PLP1
       030
       023
011567 311             RET
             BNBCD CONVERTS 16 BITS BINARY TO BCD
             BINARY NUMBER IN (H,L) REG.
             REG. C SHOULD HAVE IN IT 20 OCTAL
             RESULT IN B,D,E REG.
011570 345   BNBCD:    PUSH   H
011571 041             LXI    H,TEMP1
       274
       023
011574 006             MVI    B,03B
       003
011576 066   BCD1 :    MVI    M,00B
       000
011600 043             INX    H
011601 005             DCR    B
011602 302             JNZ    BCD1
       176
       023
011605 041   BCD2 :    LXI    H,TEMP1
       274
       023
011610 006             MVI    B,03B
       003
011612 216   BCD3 :    ADC    M
011613 047             DAA
011614 322             JNC    BCD4
       224
       023
011617 026             MVI    D,01B
       001
011621 303             JMP    BCD5
       226
       023
011624 026   BCD4 :    MVI    D,00B
```

```
       000
011626 206  BCD5 :  ADD   M
011627 047          DAA
011630 167          MOV   M,A
011631 322          JNC   BCD6
       241
       023
011634 036          MVI   E,01B
       001
011636 303          JMP   BCD7
       243
       023
011641 036  BCD6 :  MVI   E,00B
       000
011643 172  BCD7 :  MOV   A,D
011644 263          ORA   E
011645 043          INX   H
011646 005          DCR   B
011647 302          JNZ   BCD3
       212
       023
011652 341          POP   H
011653 257          XRA   A
011654 051          DAD   H
011655 345          PUSH  H
011656 015          DCR   C
011657 302          JNZ   BCD2
       205
       023
011662 341          POP   H
            RESULT IN TEMP1-TEMP3
            PUT IN B,D,E REG.
011663 041          LXI   H,TEMP3
       276
       023
011666 106          MOV   B,M
011667 053          DCX   H
011670 126          MOV   D,M
011671 053          DCX   H
011672 136          MOV   E,M
011673 311          RET
011674 000  TEMP1:  DB    0B
011675 000  TEMP2:  DB    0B
011676 000  TEMP3:  DB    0B
            FLOAT STORES CONSTANT FOR CONVERTING
            FROM INTEGER TO REAL
            FOR SAMPLE POINTS 0 TO 1024
011677 000  FLOAT:  DW    00B
       000
011701 062          DW    62B
       000
011703 144          DW    144B
       000
011705 310          DW    310B
       000
011707 220          DW    620B
```

```
        001
011711  040            DW      1440B
        003
011713  100            DW      3100B
        006
011715  200            DW      6200B
        014
011717  000            DW      14400B
        031
011721  000            DW      31000B
        062
011723  000            DW      62000B
        144
              OUT OUTPUTS TWO CHARACTERS
011725  171   OUT   :  MOV     A,C
011726  101            MOV     B,C
011727  346            ANI     360B
        360
011731  017            RRC
011732  017            RRC
011733  017            RRC
011734  017            RRC
011735  306            ADI     60B
        060
011737  117            MOV     C,A
011740  315            CALL    CO
        355
        023
011743  170   OUT1  :  MOV     A,B
011744  346            ANI     17B
        017
011746  306            ADI     60B
        060
011750  117            MOV     C,A
011751  315            CALL    CO
        355
        023
011754  311            RET
              CO PRINTS A CHARACTER ON TELE TYPE
              WHEN CALLED CHARACTER BE IN C REG.
011755  333   CO    :  IN      12B
        012
011757  346            ANI     200B
        200
011761  312            JZ      CO
        355
        023
011764  171            MOV     A,C
011765  323            OUT     13B
        013
011767  311            RET
              CRLF FOR LINE SPACING
011770  016   CRLF  :  MVI     C,15B
        015
011772  315            CALL    CO
        355
```

```
         001
011711   040        DW     1440B
         003
011713   100        DW     3100B
         006
011715   200        DW     6200B
         014
011717   000        DW     14400B
         031
011721   000        DW     31000B
         062
011723   000        DW     62000B
         144
              OUT OUTPUTS TWO CHARACTERS
011725   171  OUT  :  MOV   A,C
011726   101          MOV   B,C
011727   346          ANI   360B
         360
011731   017          RRC
011732   017          RRC
011733   017          RRC
011734   017          RRC
011735   306          ADI   60B
         060
011737   117          MOV   C,A
011740   315          CALL  CO
         355
         023
011743   170  OUT1 :  MOV   A,B
011744   346          ANI   17B
         017
011746   306          ADI   60B
         060
011750   117          MOV   C,A
011751   315          CALL  CO
         355
         023
011754   311          RET
              CO PRINTS A CHARACTER ON TELE TYPE
              WHEN CALLED CHARACTER BE IN C REG.
011755   333  CO   :  IN    12B
         012
011757   346          ANI   200B
         200
011761   312          JZ    CO
         355
         023
011764   171          MOV   A,C
011765   323          OUT   13B
         013
011767   311          RET
              CRLF FOR LINE SPACING
011770   016  CRLF :  MVI   C,15B
         015
011772   315          CALL  CO
         355
```

```
       023
011775 016          MVI   C,12B
       012
011777 315          CALL  CO
       355
       023
012002 311          RET
012010              ORG   12010B
          MAIN PROGRAM STARTS
012010 061          LXI   SP,27777B   ;INIALIZE ZTACK POINTER
       377
       057
          NUMBER OF SAMPLE POINTS ENTERED IN (H,L)REG
012013 041          LXI   H,200B
       200
       000
          FOR 128 SAMPLE POINTS
          LOWER BYTE IN LOCATION 12014
          HIGHER BYTE IN LOCATION 12015
012016 042          SHLD  N
       012
       016
012021 041          LXI   H,XREAL
       060
       024
012024 315          CALL  ZEROM  ;FOR MAKING LSB PART OF DATA ZERO
       243
       022
012027 041          LXI   H,XIMAG
       060
       034
012032 315          CALL  ZEROM
       243
       022
012035 041          LXI   H,XREAL
       060
       024
012040 315          CALL  READ   ;FOR READING REAL DATA FROM TAPE
       273
       022
012043 041          LXI   H,XIMAG
       060
       034
012046 315          CALL  READ
PAGE 11

       273

       022
012051 315          CALL  FFTIN  ;FOR FFT
       000
       014
012054 315          CALL  INTFP   ;FOR INTEGER TO REAL CONVERSION
```

```
       353
       Ø22
              AND PRINTING ON TELE TYPE
Ø12Ø57 166            HLT
              RESERVE LOCATIONS FOR 1Ø24 SAMPLE POINTS
Ø12Ø6Ø        XREAL:  DS     4ØØØB
Ø16Ø6Ø        XIMAG:  DS     4ØØØB
                      END
NO OF ERRORS:  ØØ
```

Date Slip A 55283

This book is to be returned on the date last stamped.

CD 6.72.9